Clustering

Fabrice Rossi

CEREMADE
Université Paris Dauphine

2021

Outline

Unsupervised learning

Setting

- $\mathcal{D} = ((\mathbf{X}_i)_{1 \leq i \leq N})$
- no target value!
- goal: “understanding” the data
- in practice, many concrete goals such as
  - finding clusters
  - finding frequent patterns
  - finding outliers
  - modeling the data distribution
  - etc.

Unsupervised learning

Setting

- $\mathcal{D} = ((\mathbf{X}_i)_{1 \leq i \leq N})$
- no target value!
- goal: “understanding” the data
- in practice, many concrete goals such as
  - finding clusters
  - finding frequent patterns
  - finding outliers
  - modeling the data distribution
  - etc.

Clusters

Definition

In a data set, a *cluster* is a group of objects that are more similar to each other than to objects from the rest of the data set.

Clusters

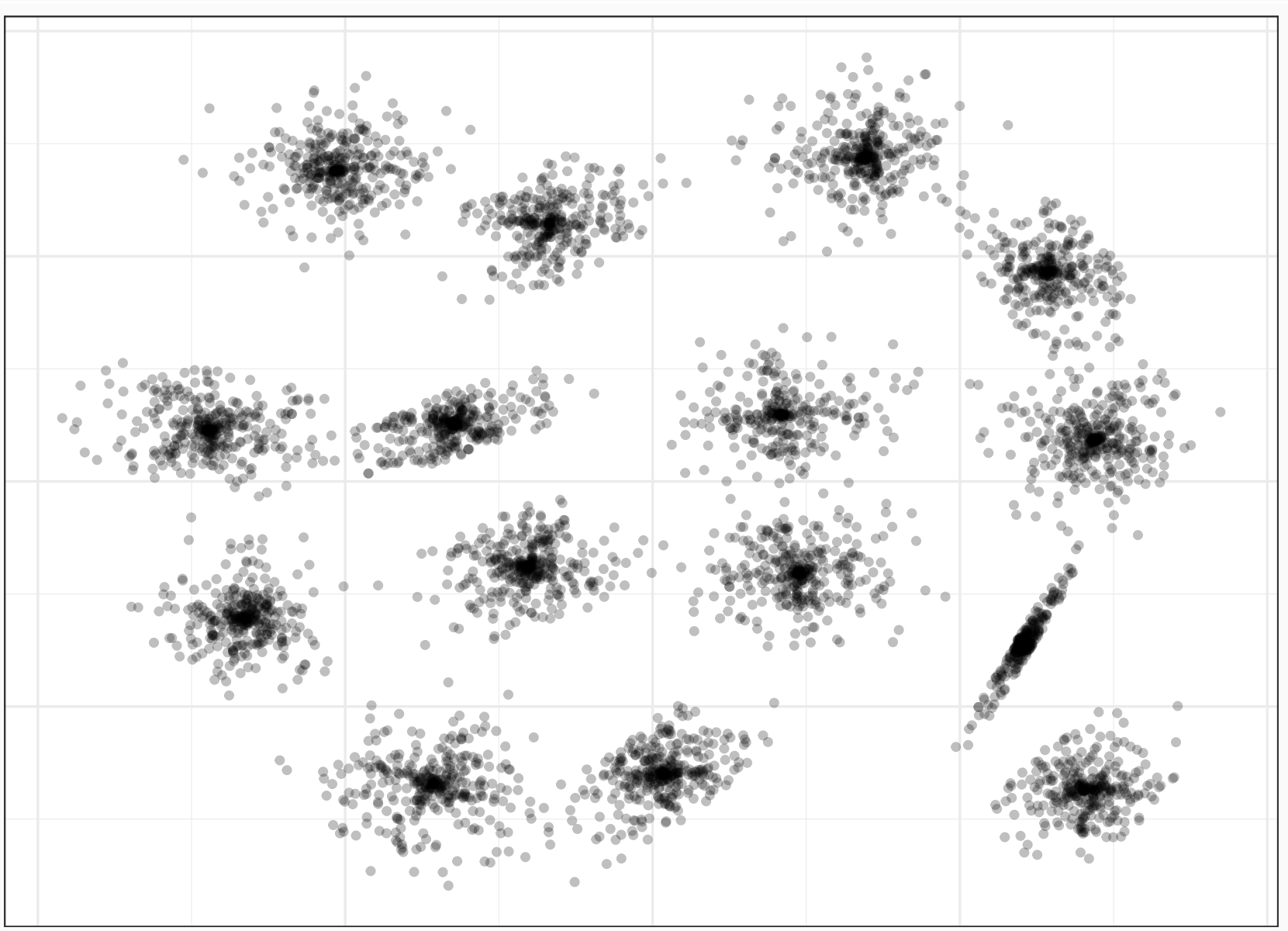
Definition

In a data set, a *cluster* is a group of objects that are more similar to each other than to objects from the rest of the data set.

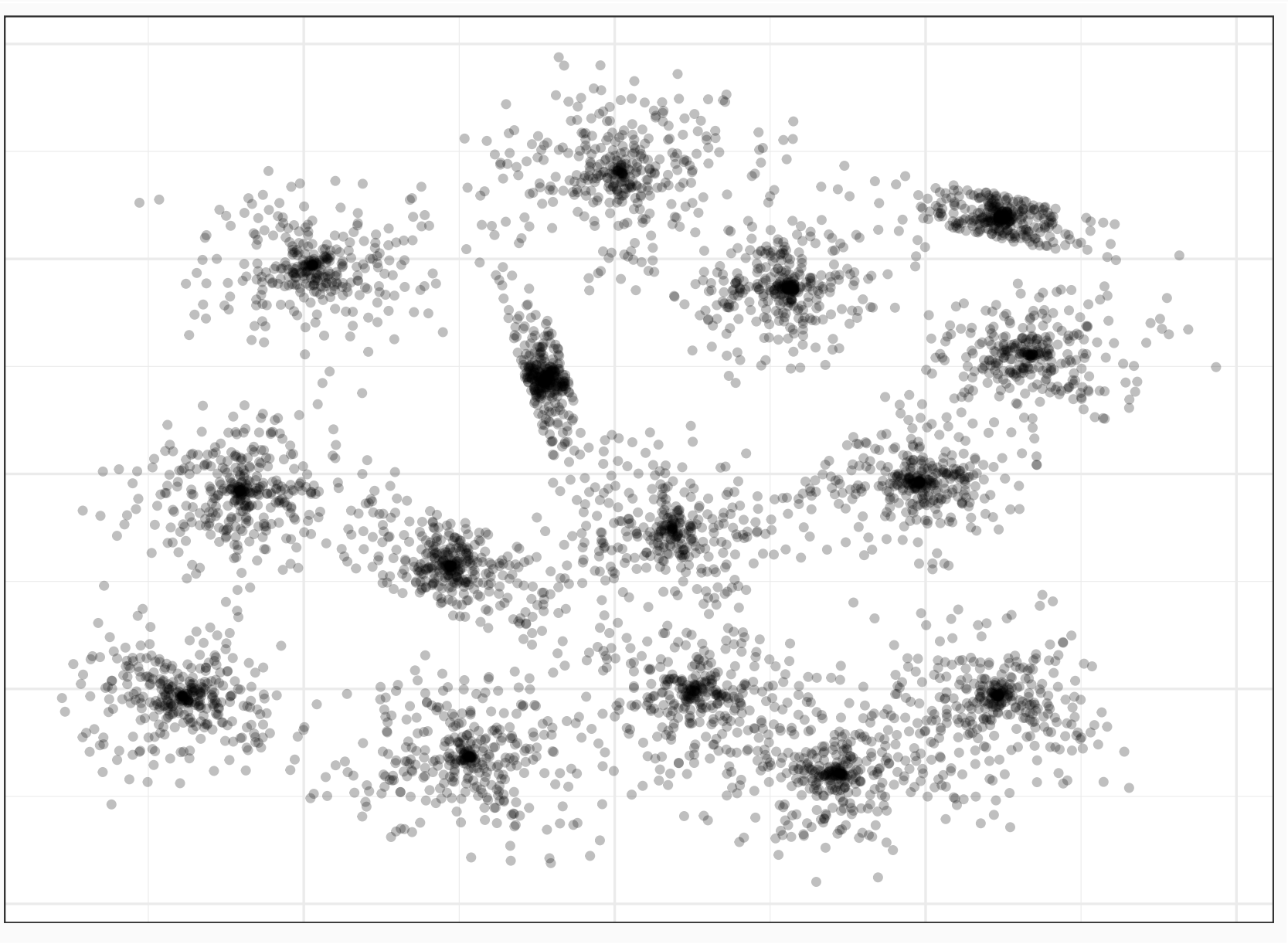
Difficulties

- group?
- similar?
- more than?

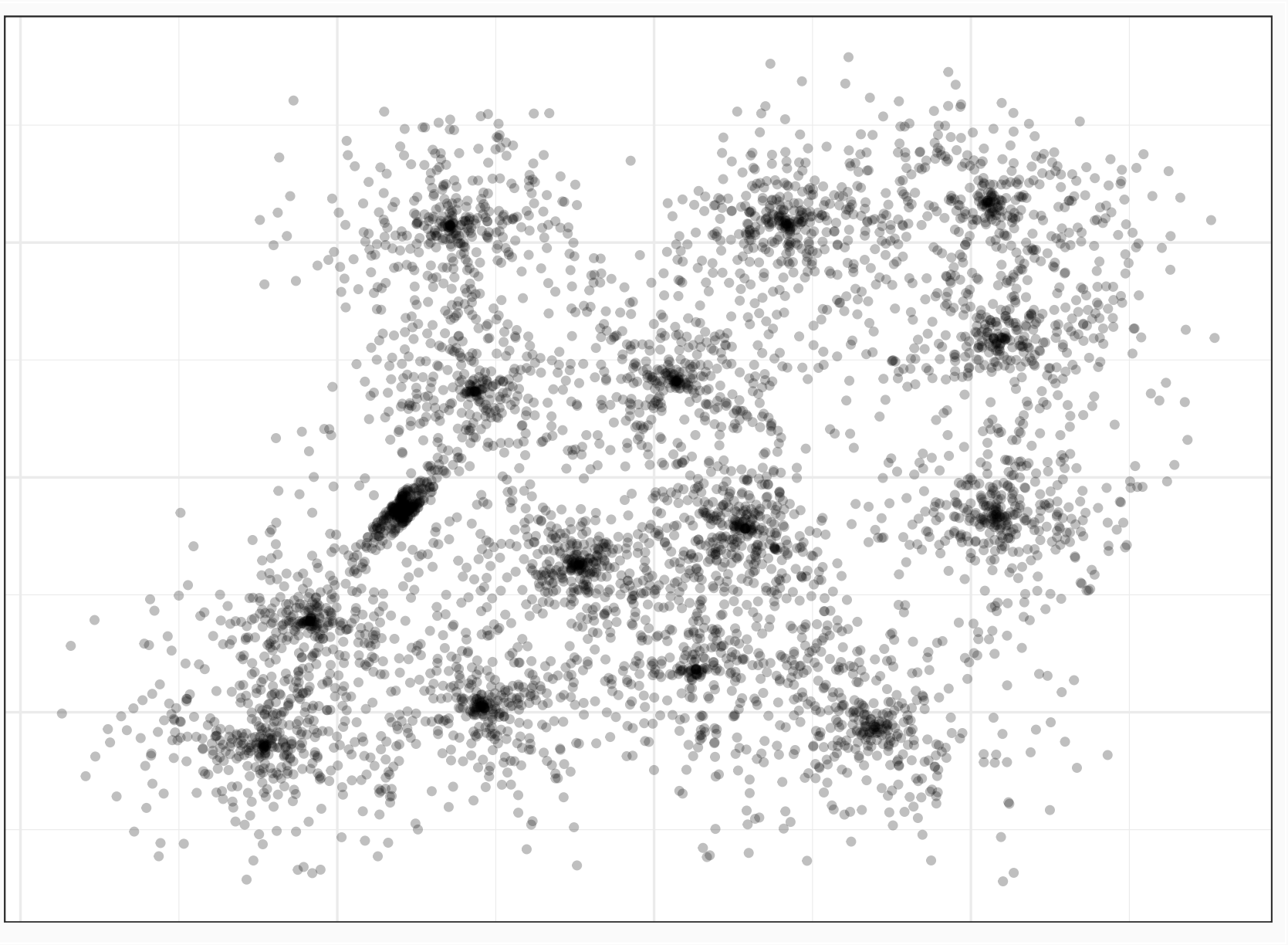
Non obvious task



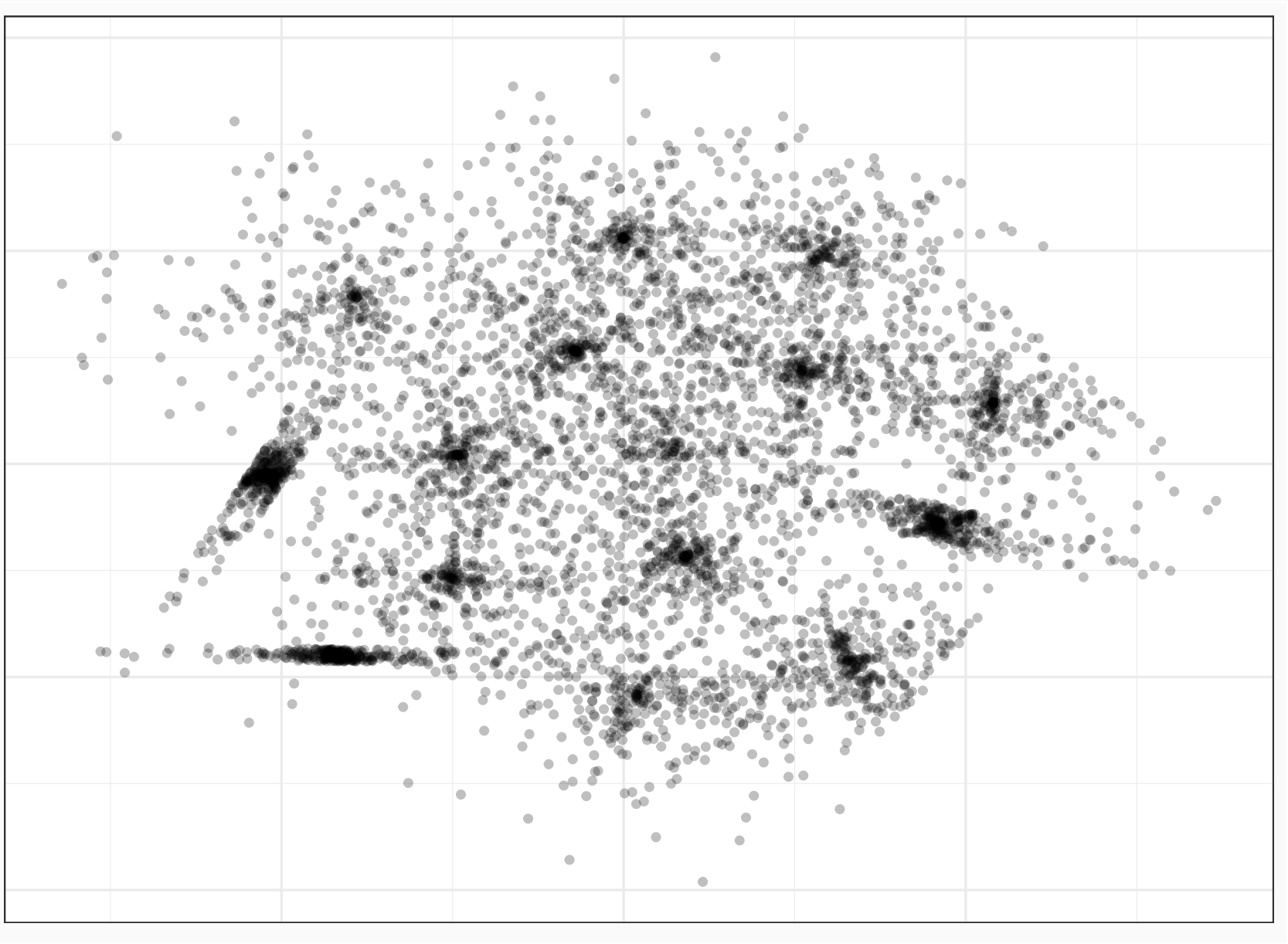
Non obvious task



Non obvious task



Non obvious task



Applications

Simplification (pre-processing)

- large scale data analysis: replace a group of objects by a single typical object
- coarse grain analysis before a finer grain one (see e.g. Yippy)

Applications

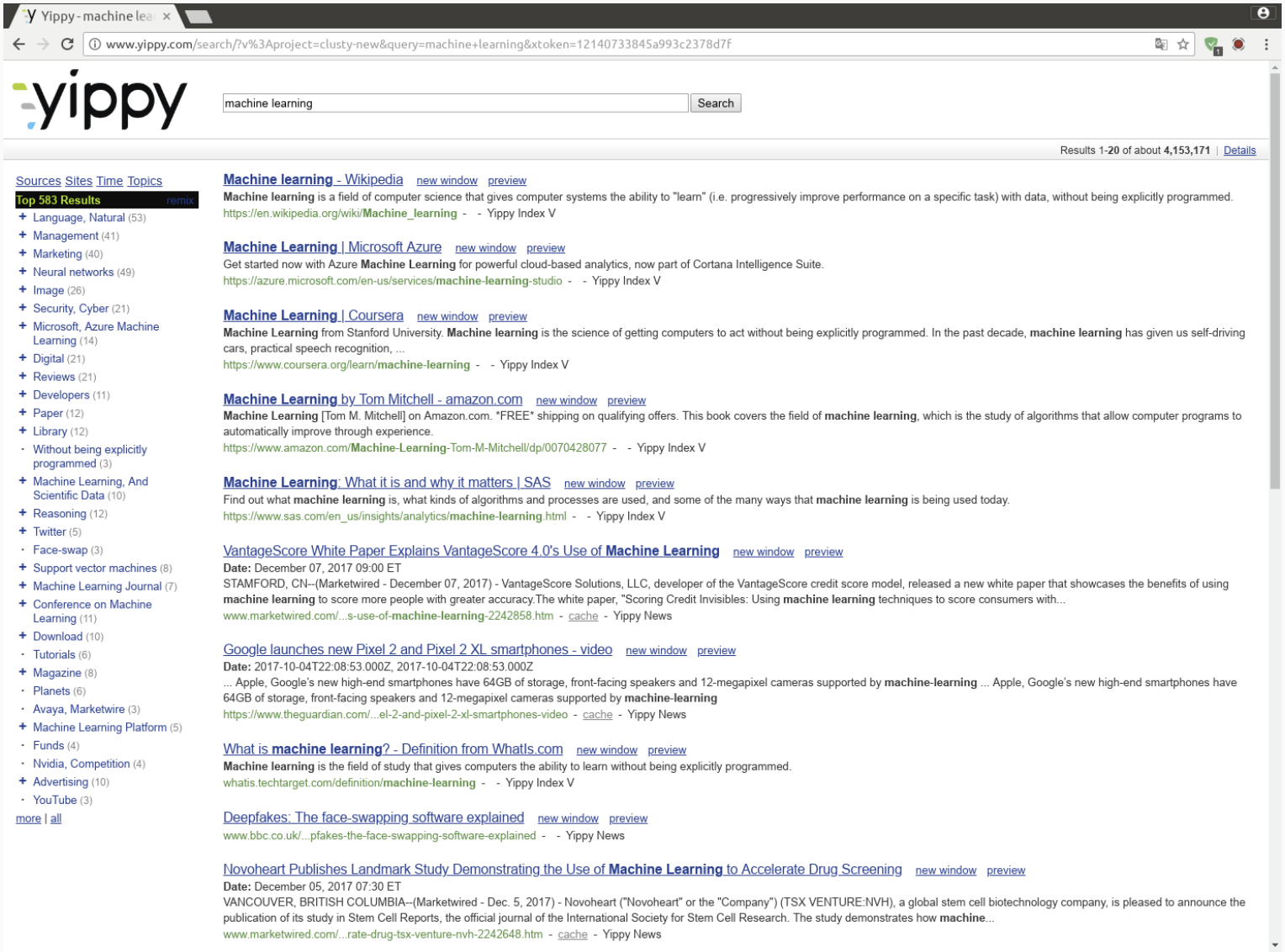
Simplification (pre-processing)

- large scale data analysis: replace a group of objects by a single typical object
- coarse grain analysis before a finer grain one (see e.g. Yippy)

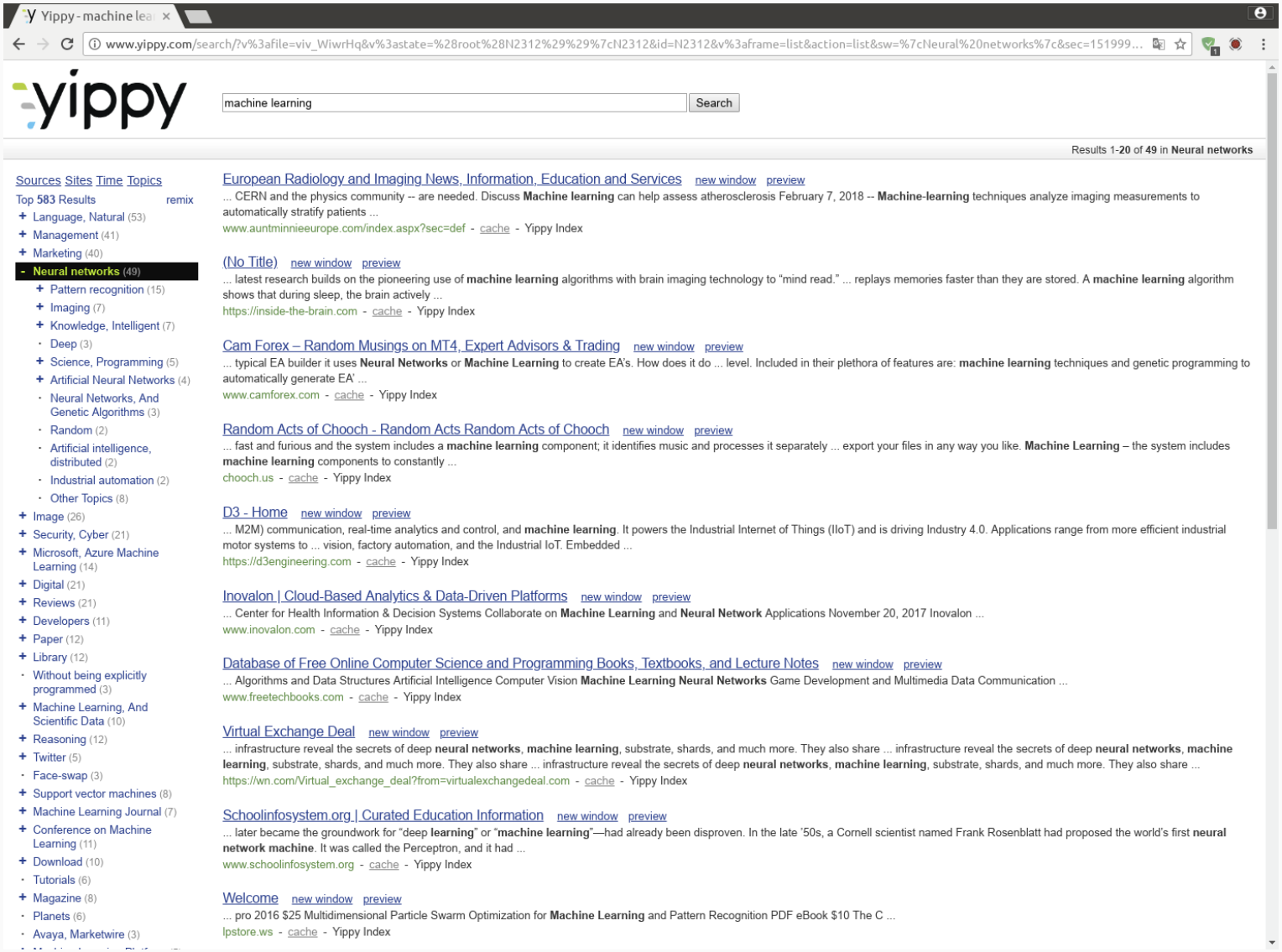
Knowledge discovery

- consumer analysis
- image analysis (zone extraction)
- evolutionary biology
- etc.

Yippy example



Yippy example



Yippy example

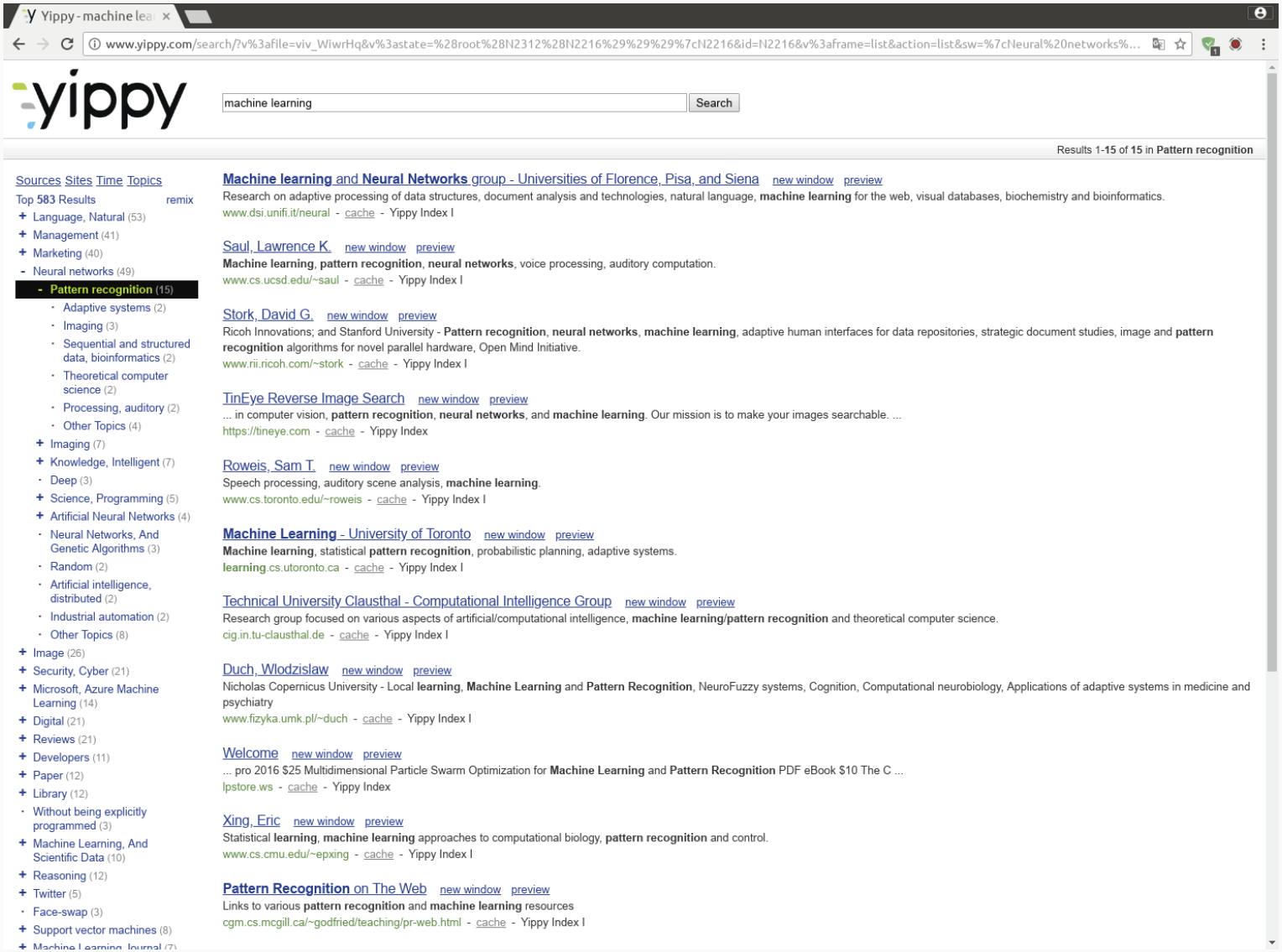


Image quantization

Image as data set

- a pixel: a vector in $\mathbb{R}^3$
- image: set of pixels a.k.a. vectors
- clustering: comparable pixels

Image quantization

Pixels

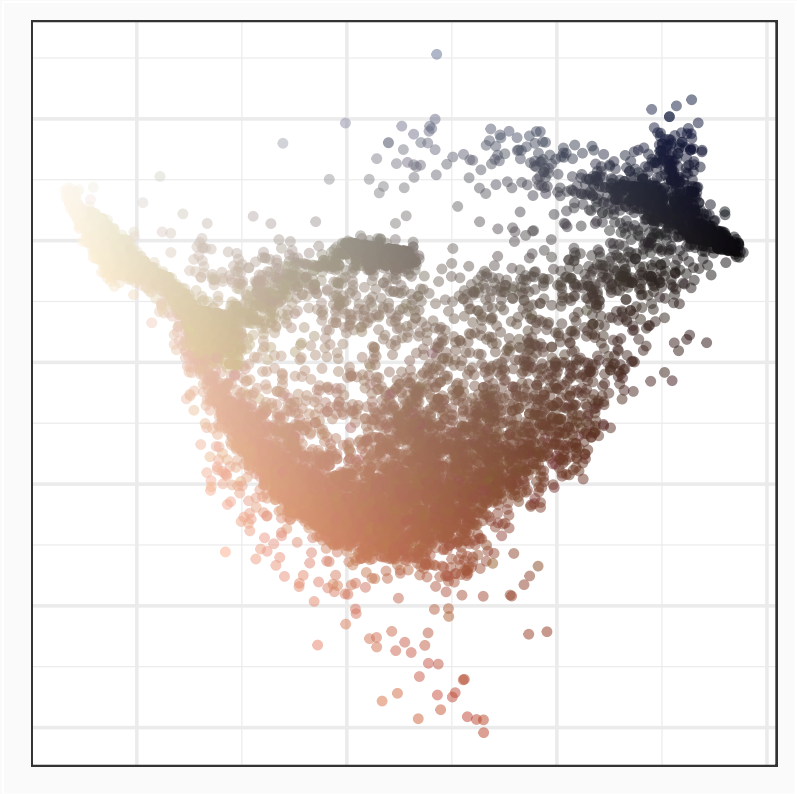


Image quantization

50 typical pixels

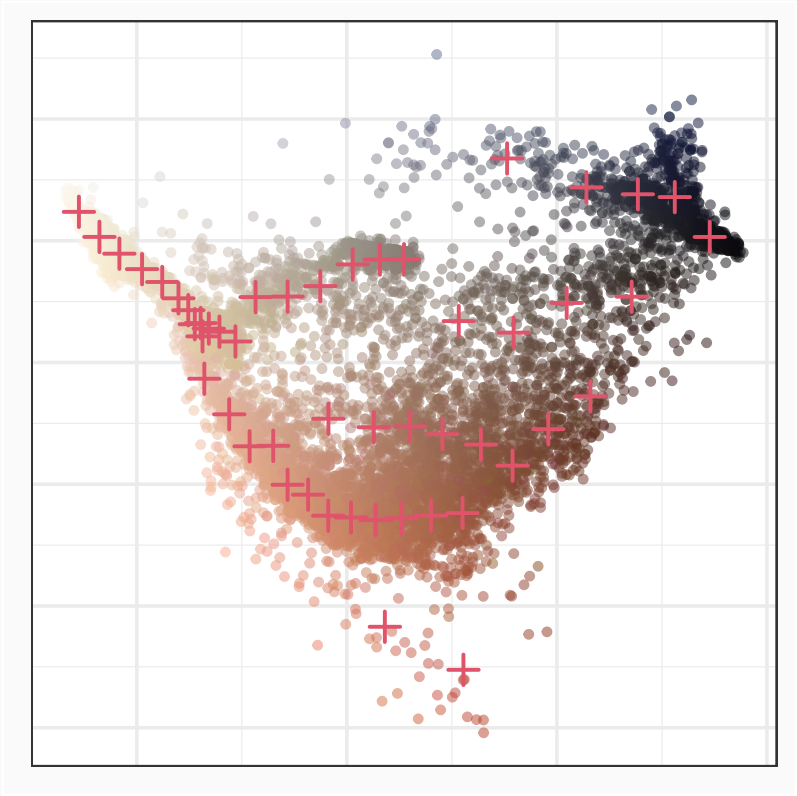
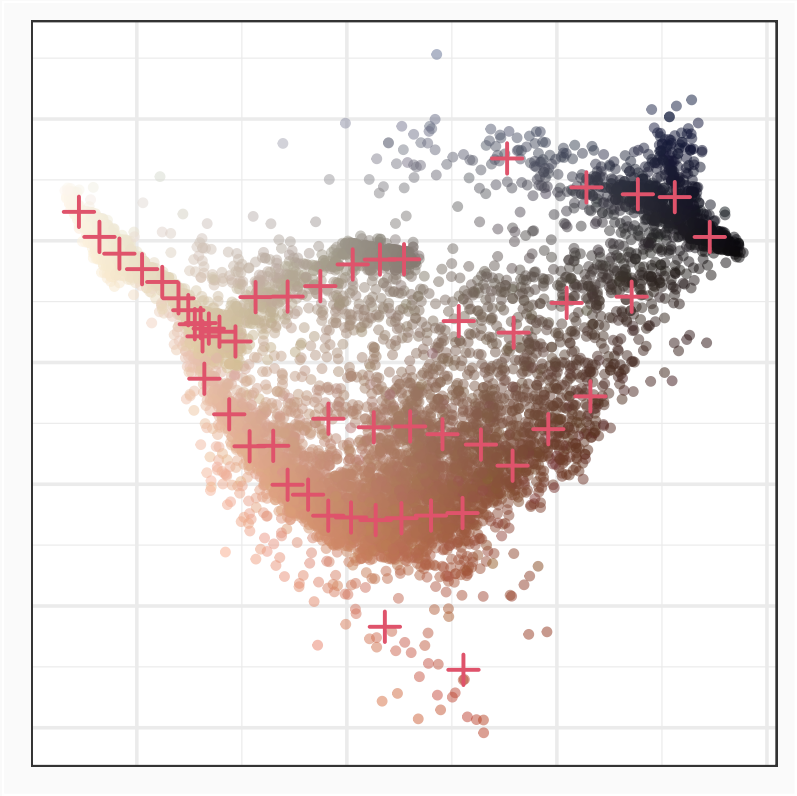


Image quantization

50 typical pixels



How to build a clustering method?

Ingredients

- similarity or dissimilarity: how to compare objects?
- group structure: set, “fuzzy” set, probabilistic membership?
- clustering structure: disjoint sets, overlapping sets, complete partition?

Classical example: standard k-means

- dissimilarity: euclidean distance on numerical variables (i.e. $\mathcal{X} = \mathbb{R}^P$)
- groups: standard sets
- clustering structure: groups for a partition of the data set

Clustering is ill posed

Problems

- vague definition with possible extensions (e.g. find groups that reflect the “underlying structure” of the data set)
- vastly different practical goals (from pre-processing to knowledge discovery)
- no universal quality criterion:
  - almost one criterion per method!
  - lack of task oriented criterion
- impossibility result in some situations

Clustering is ill posed

Problems

- vague definition with possible extensions (e.g. find groups that reflect the “underlying structure” of the data set)
- vastly different practical goals (from pre-processing to knowledge discovery)
- no universal quality criterion:
  - almost one criterion per method!
  - lack of task oriented criterion
- impossibility result in some situations

Clustering as a process

- for knowledge discovery
- usefulness rather than quality

Results and tools

Using a clustering

- cluster analysis: what do they contain?
  - list of the members
  - representative element(s)
- cluster “positioning”:
  - relative positioning in space
  - significant differences

Tasks oriented analysis

- can the analyst understand the clusters?
- can the clustering summarize the data? In what sense?

Families of clustering algorithms

Four main families...

- hierarchical algorithms: hierarchical clustering
- centroid based algorithms: k-means
- density based algorithms: DBSCAN
- probabilistic algorithms: mixture models and EM

Families of clustering algorithms

Four main families...

- hierarchical algorithms: hierarchical clustering
- centroid based algorithms: k-means
- density based algorithms: DBSCAN
- probabilistic algorithms: mixture models and EM

but a complex landscape

- k-means as a limit case of some mixture models
- DBSCAN is somewhat related to single linkage hierarchical clustering
- numerous hybrid techniques
- etc.

Clustering and dissimilarity

Minimal assumption

- $\mathcal{X}$ is equipped with a dissimilarity d
- d is a dissimilarity on $\mathcal{X}$ iff:
  1. d is a function from $\mathcal{X} \times \mathcal{X}$ to $\mathbb{R}^+$
  2. $\forall \mathbf{X}, \mathbf{X}',\ d(\mathbf{X}, \mathbf{X}') = d(\mathbf{X}', \mathbf{X})$
  3. $\forall \mathbf{X}, \mathbf{X}',\ \mathbf{X} \neq \mathbf{X}' \Leftrightarrow d(\mathbf{X}, \mathbf{X}') > 0$

Links with clustering

- many algorithms can work with arbitrary dissimilarities
- density, separability, compactness, in general quality metrics are expressed in terms of the dissimilarity
- results **strongly** depend on the dissimilarity
- plain $\mathbb{R}^P$: euclidean dissimilarity!

Outline

Partitions

Clustering and partition

$\mathcal{D} = ((\mathbf{X}_i)_{1 \leq i \leq N})$

- group: a subset of $\{1, \ldots, N\}$
- clustering structure: each observation (i.e. index in $\{1, \ldots, N\}$) is in one unique group + no empty group
- building a clustering $\Leftrightarrow$ choosing a partition!
- *here* a *cluster* is also a *class*

Partitions

Clustering and partition

$\mathcal{D} = ((\mathbf{X}_i)_{1 \leq i \leq N})$

- group: a subset of $\{1, \ldots, N\}$
- clustering structure: each observation (i.e. index in $\{1, \ldots, N\}$) is in one unique group + no empty group
- building a clustering $\Leftrightarrow$ choosing a partition!
- *here* a *cluster* is also a *class*

Remarks

- this is only *one* possible structure among others!
- one can have $\mathbf{X}_i = \mathbf{X}_j$ for $i \neq j$
  - we expect i and j to be in the same group!
  - groups are described at the index level

Hierarchy

Partial order on partitions

- a partition P is finer than a partition Q ($P \leq Q$) if any class of P is a subset of a class of Q
- example

$$P = \{\{1,2\},\{3,4\},\{5,6\}\} \leq \{\{1,2,3,4\},\{5,6\}\} = Q$$

Hierarchy

A hierarchy for $\mathcal{D} = ((\mathbf{X}_i)_{1 \leq i \leq N})$ is a fully ordered set of partitions containing the trivial partitions

1. $\{\{1,\ldots,N\}\}$
2. $\{\{1\},\{2\},\ldots,\{N\}\}$

Dendrogram

Graphical representation of a hierarchy by a tree

Dendrogram

Graphical representation of a hierarchy by a tree

A B C D E F

Dendrogram

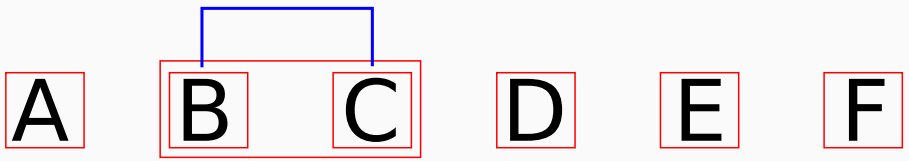
Graphical representation of a hierarchy by a tree

- most refined partition (trivial)

Dendrogram

Graphical representation of a hierarchy by a tree

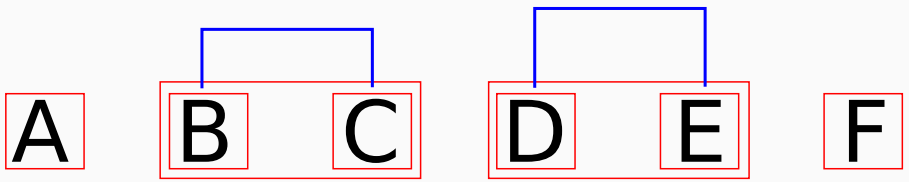
- most refined partition (trivial)
- B and C are in the same class at level 2: leaves of a common node



Dendrogram

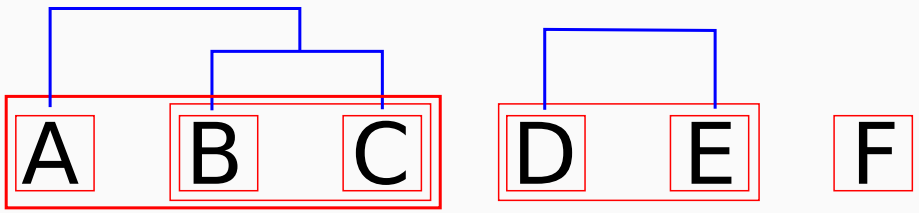
Graphical representation of a hierarchy by a tree

- most refined partition (trivial)
- B and C are in the same class at level 2: leaves of a common node
- same thing for D and E, but a level 3



Dendrogram

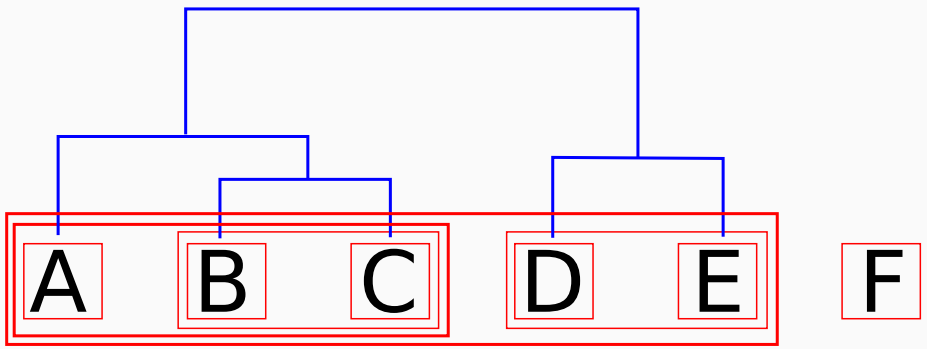
Graphical representation of a hierarchy by a tree



- most refined partition (trivial)
- B and C are in the same class at level 2: leaves of a common node
- same thing for D and E, but a level 3
- A, B, and C: new node that shows the order between partitions

Dendrogram

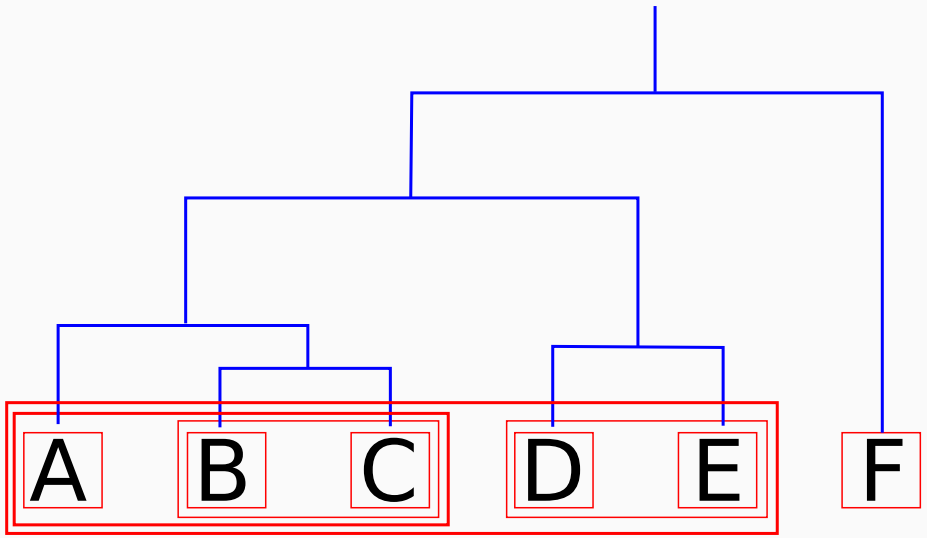
Graphical representation of a hierarchy by a tree



- most refined partition (trivial)
- B and C are in the same class at level 2: leaves of a common node
- same thing for D and E, but a level 3
- A, B, and C: new node that shows the order between partitions
- more nodes until the coarsest partition

Dendrogram

Graphical representation of a hierarchy by a tree

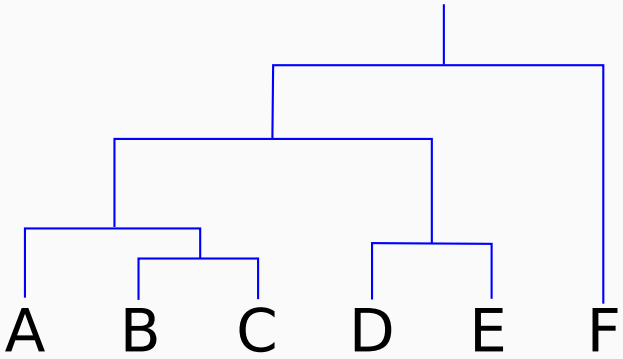


- most refined partition (trivial)
- B and C are in the same class at level 2: leaves of a common node
- same thing for D and E, but a level 3
- A, B, and C: new node that shows the order between partitions
- more nodes until the coarsest partition

Dendrogram

Understanding dendrograms

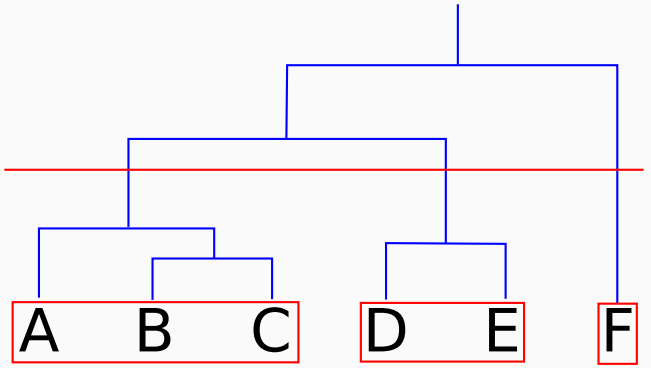
- provides a summary of the hierarchy
- a level in the tree (a.k.a. a node) corresponds to merging two classes
- the height of a level/node gives the "quality" of the merging



Dendrogram

Understanding dendrograms

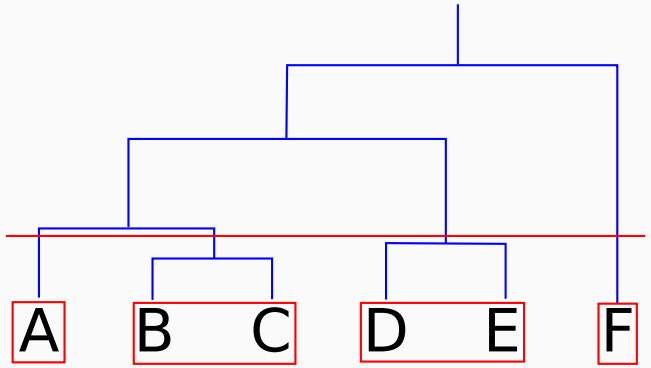
- provides a summary of the hierarchy
- a level in the tree (a.k.a. a node) corresponds to merging two classes
- the height of a level/node gives the "quality" of the merging
- horizontal cut $\Rightarrow$ partition



Dendrogram

Understanding dendrograms

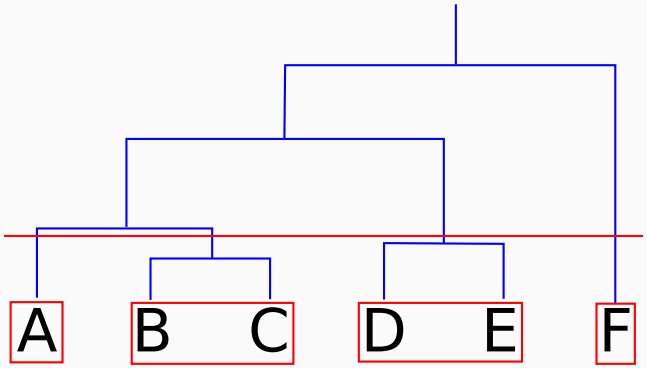
- provides a summary of the hierarchy
- a level in the tree (a.k.a. a node) corresponds to merging two classes
- the height of a level/node gives the "quality" of the merging
- horizontal cut $\Rightarrow$ partition



Dendrogram

Understanding dendrograms

- provides a summary of the hierarchy
- a level in the tree (a.k.a. a node) corresponds to merging two classes
- the height of a level/node gives the "quality" of the merging
- horizontal cut $\Rightarrow$ partition



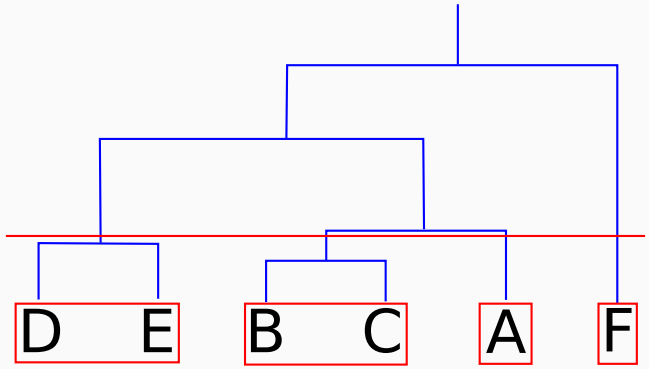
Common errors

- the dendrogram summarizes the **hierarchy** not the data
- do not interpret the order of the leaves: it is almost arbitrary (2^{N-2} orders for N leaves)

Dendrogram

Understanding dendrograms

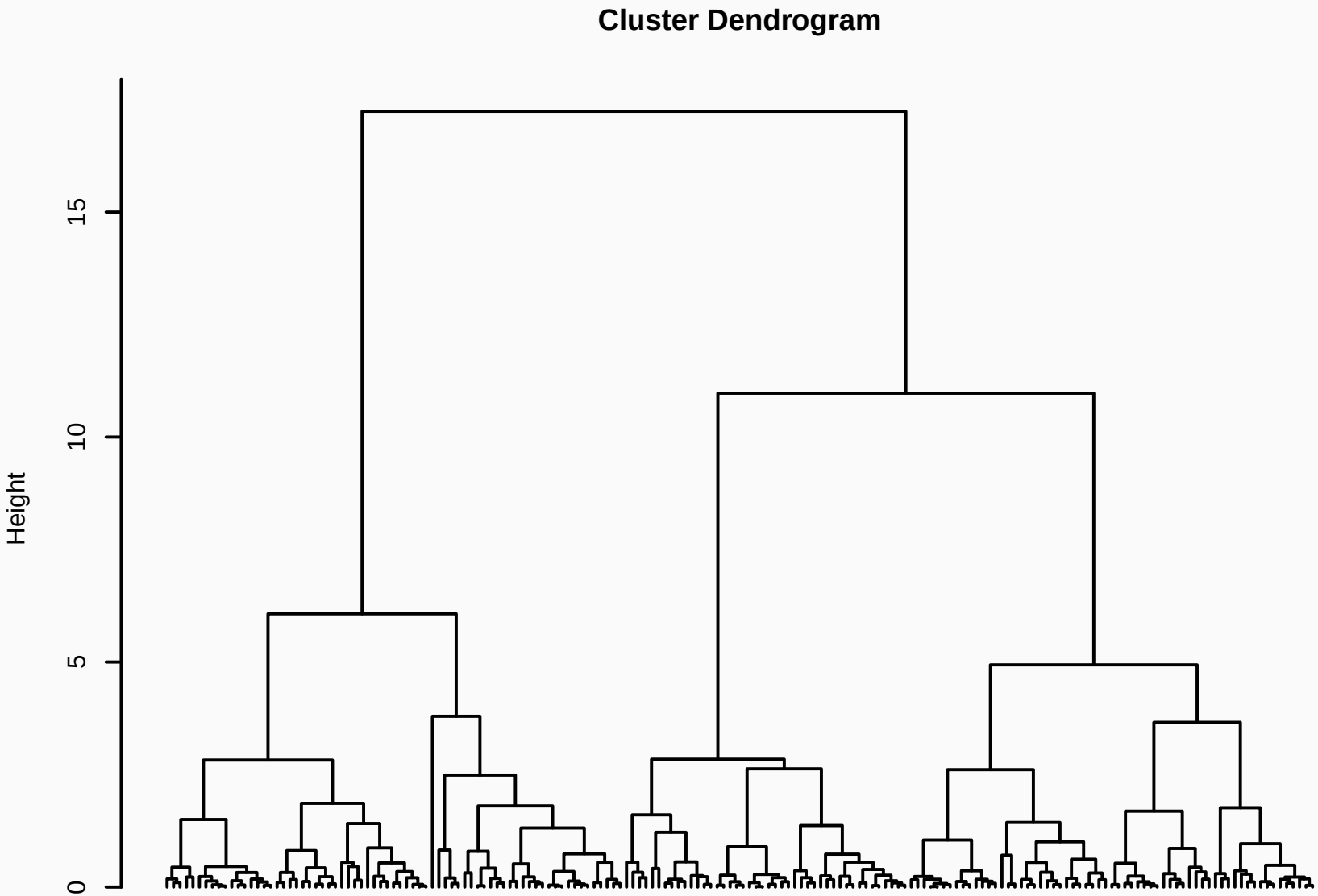
- provides a summary of the hierarchy
- a level in the tree (a.k.a. a node) corresponds to merging two classes
- the height of a level/node gives the "quality" of the merging
- horizontal cut $\Rightarrow$ partition



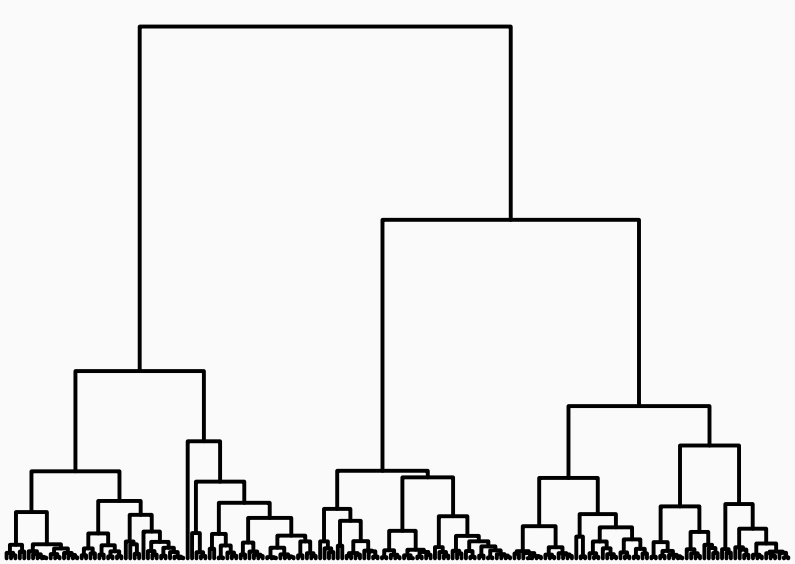
Common errors

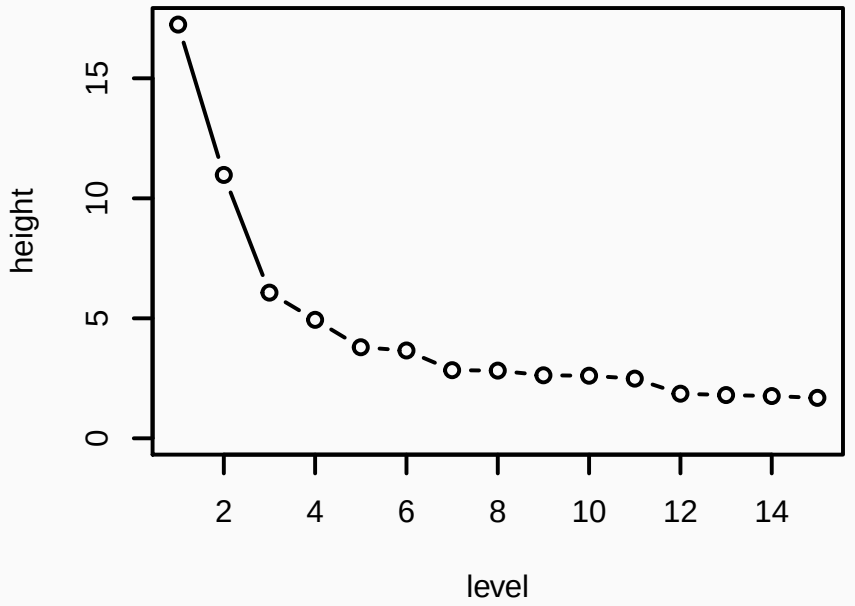
- the dendrogram summarizes the **hierarchy** not the data
- do not interpret the order of the leaves: it is almost arbitrary (2^{N-2} orders for N leaves)

Example



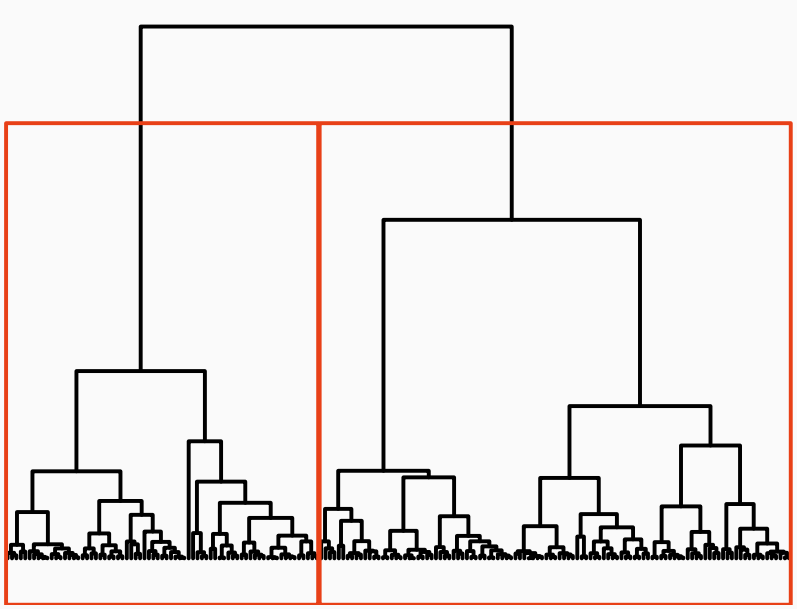
Example

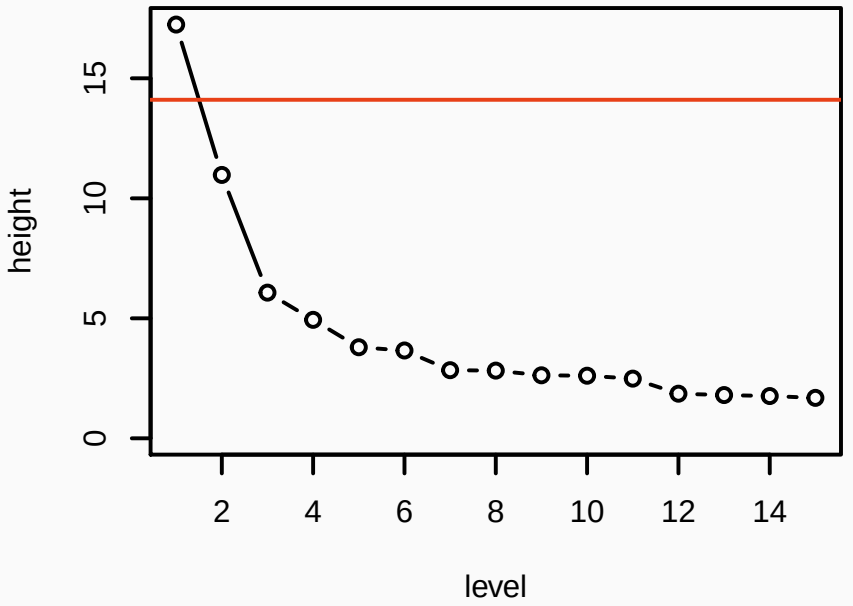




- look for “gaps” between levels: potential candidates for interesting partitions
- local partitions (i.e. branches) might also be interesting

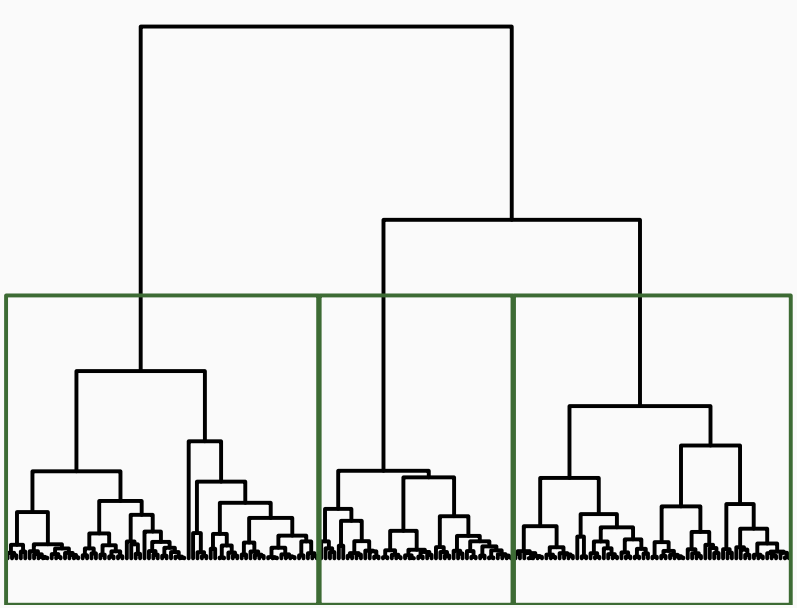
Example

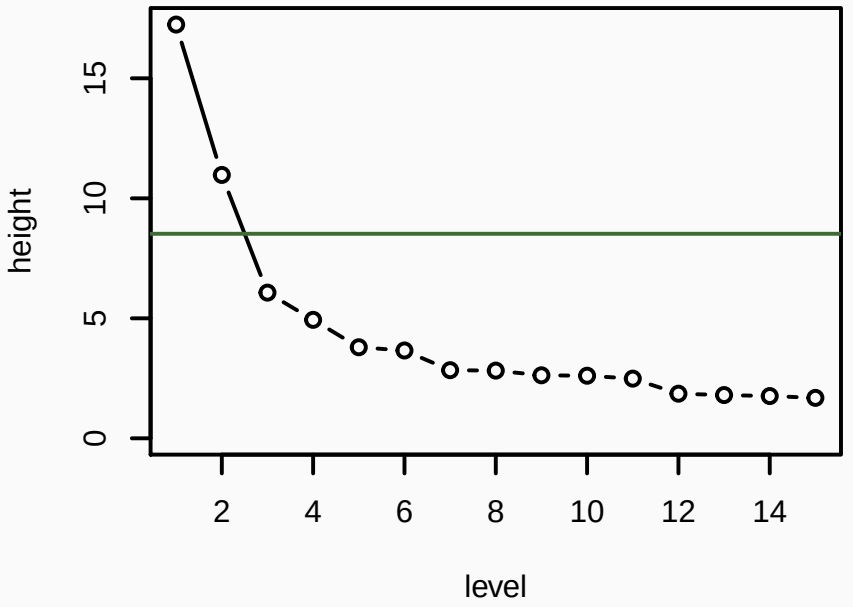




- look for “gaps” between levels: potential candidates for interesting partitions
- local partitions (i.e. branches) might also be interesting

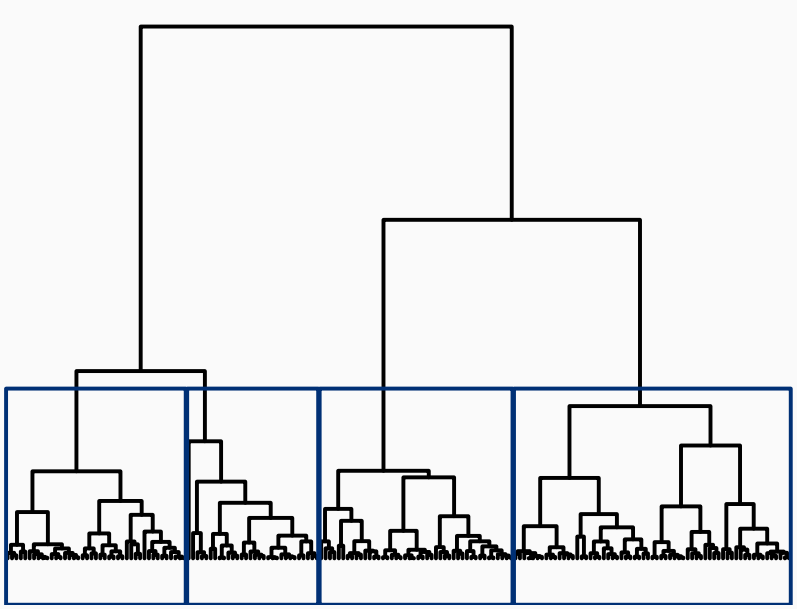
Example

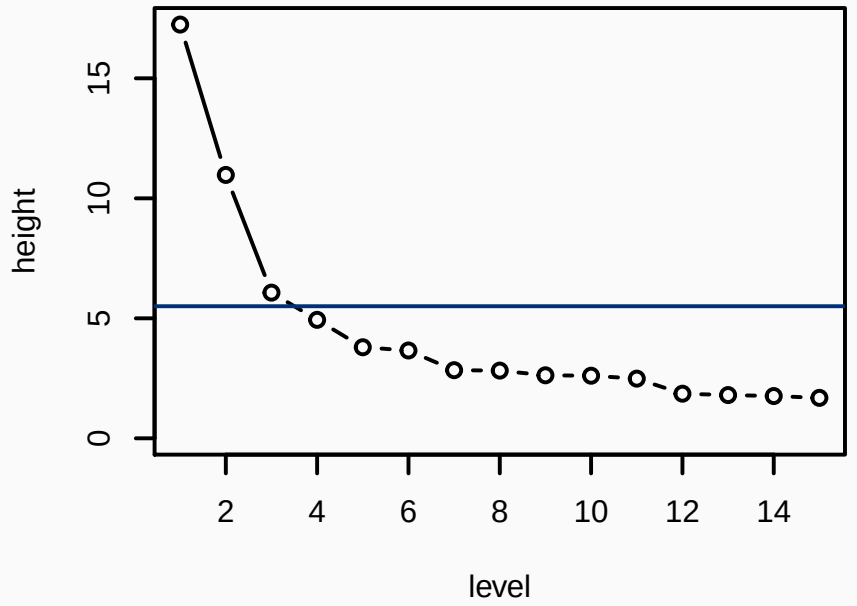




- look for “gaps” between levels: potential candidates for interesting partitions
- local partitions (i.e. branches) might also be interesting

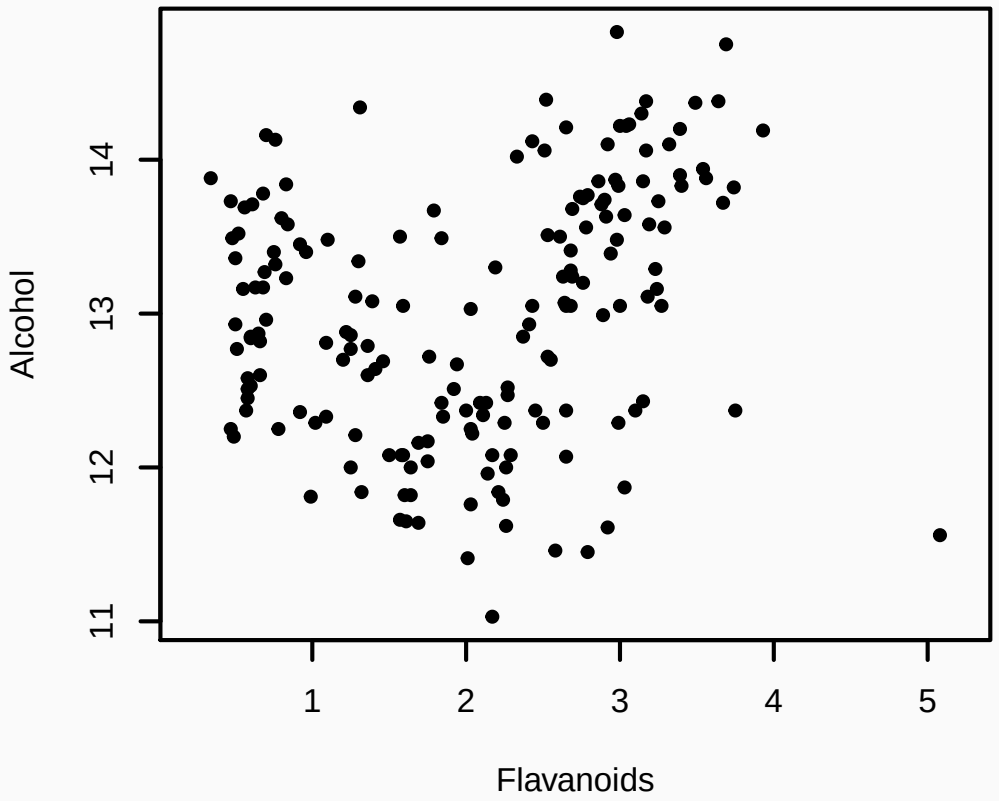
Example

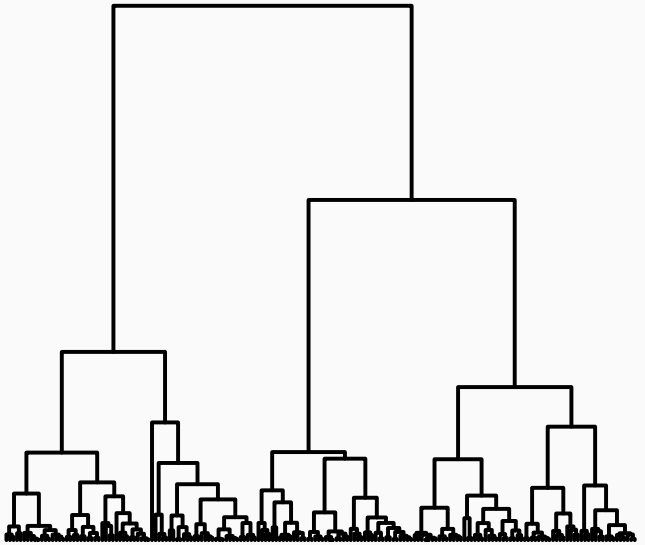




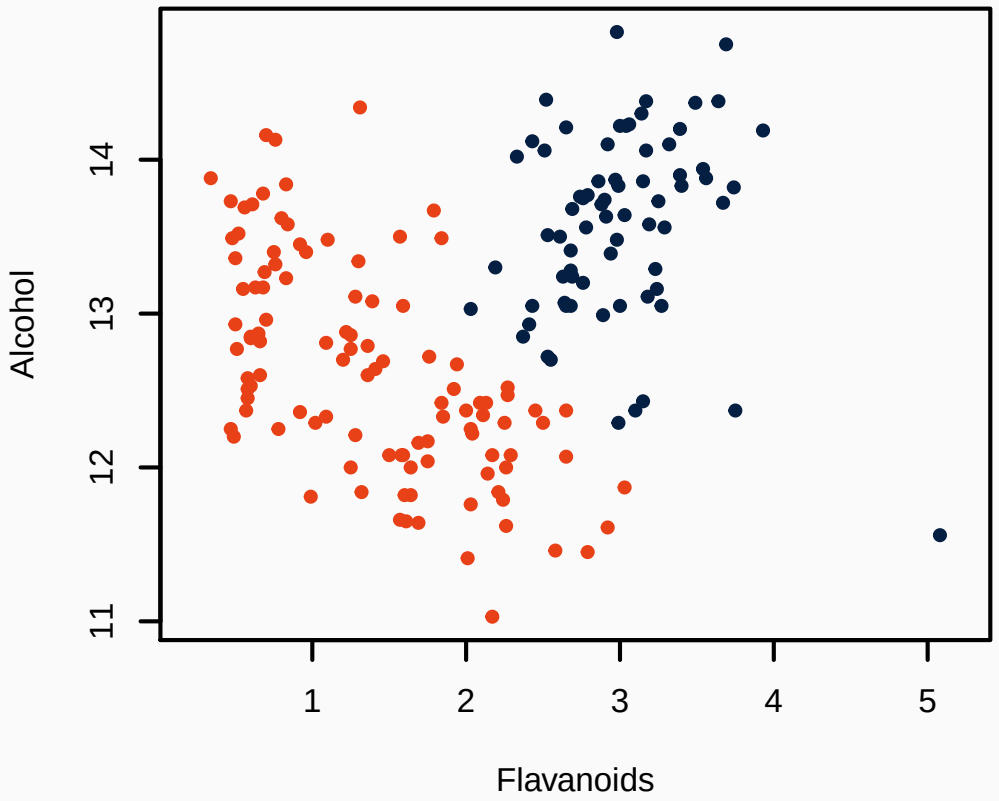
- look for “gaps” between levels: potential candidates for interesting partitions
- local partitions (i.e. branches) might also be interesting

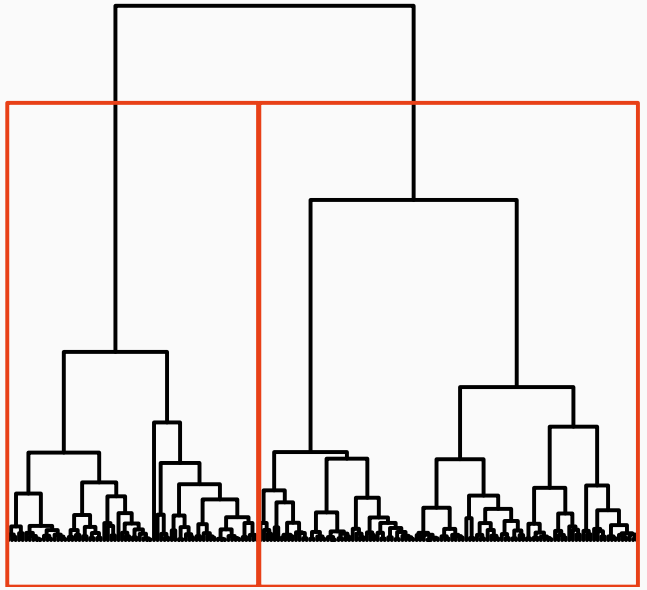
Example



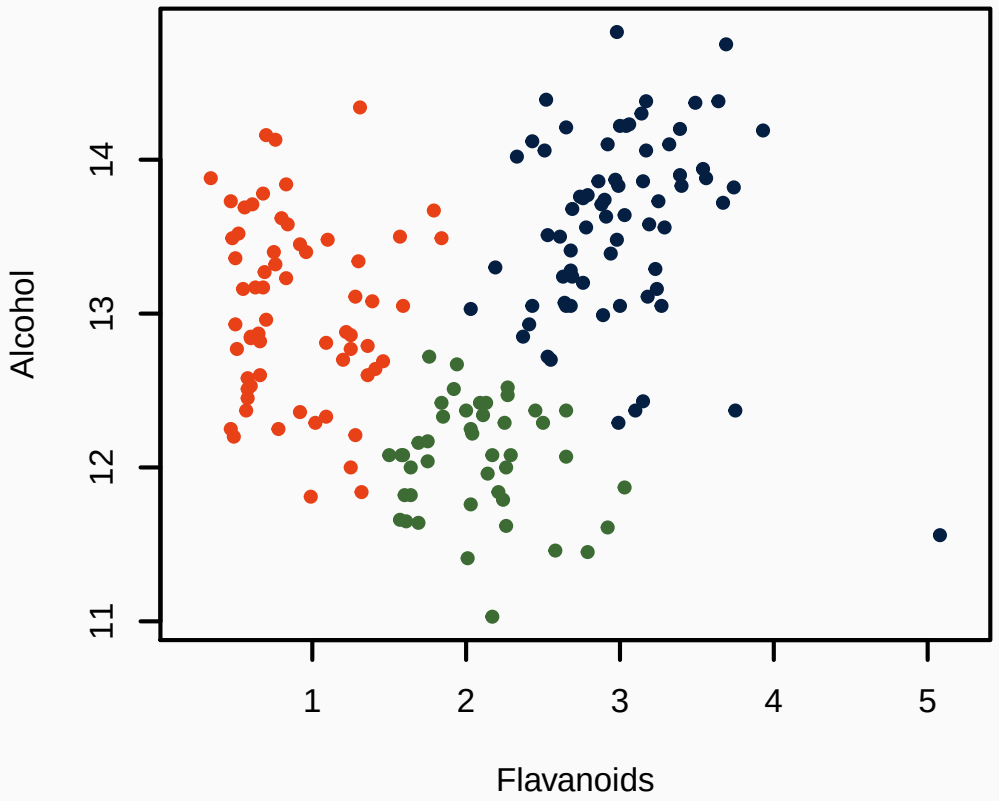


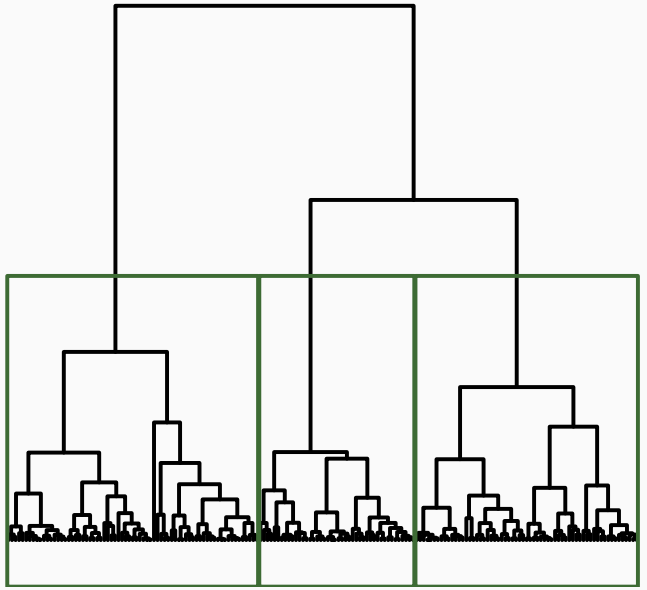
Example



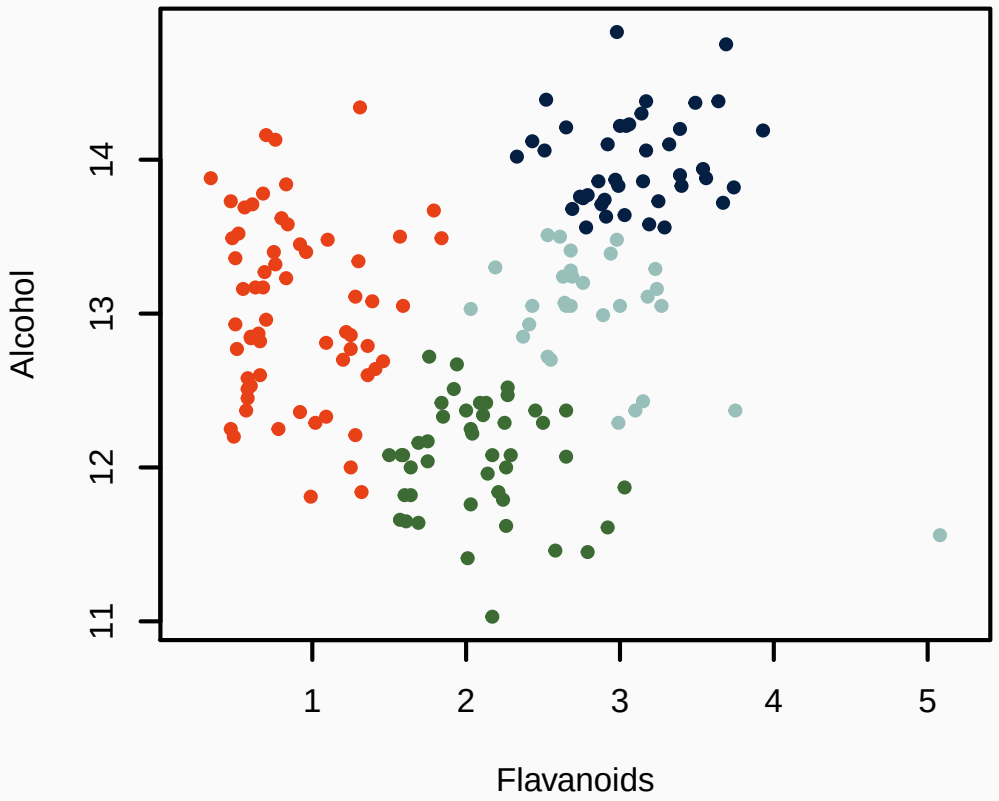


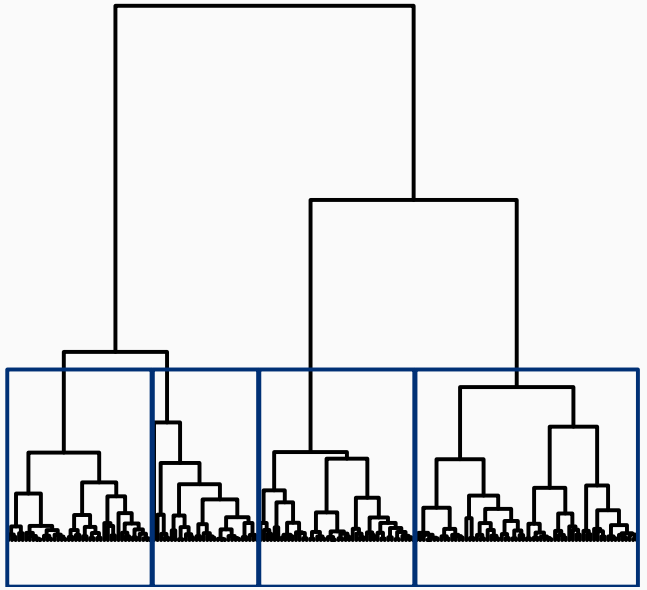
Example





Example





Hierarchical clustering

Agglomerative methods

- start with $P = \{\{1\}, \{2\}, \ldots, \{N\}\}$
- merge two classes in such a way that the resulting class is a *cluster*
- keep doing that until reaching $Q = \{\{1, \ldots, N\}\}$

Divisive methods

- start with $Q = \{\{1, \ldots, N\}\}$
- split one class into two sub-classes that are *clusters*
- keep doing that until reaching $P = \{\{1\}, \{2\}, \ldots, \{N\}\}$

Hierarchical clustering

Common elements

- those methods produce a hierarchy with N partitions
- they need a way to assess suitability of the classes as clusters
- dissimilarity based algorithms

Differences

- numerous agglomerative methods
- relatively few divisive ones
- very different computational problems:
  - agglomerative: $\Theta\left(N^2\right)$ potential merges at each step
  - divisive: $\Theta\left(2^{N-1}\right)$ possible splits at the first step

Agglomerative methods

Core principle

- merge classes when they contain *similar* objects
- for singleton classes $\{\mathbf{X}_i\}$ and $\{\mathbf{X}_j\}$ use $d(\mathbf{X}_i, \mathbf{X}_j)$ to judge similarity
- key point: extend the dissimilarity to groups of objects

General algorithm

- initial partition: $\mathcal{P}^1 = \{\{1\}, \{2\}, \ldots, \{N\}\}$
- for k from 2 to N:
  - compute the dissimilarity between all current classes in $\mathcal{P}^{k-1}$
  - build $\mathcal{P}^k$ from $\mathcal{P}^{k-1}$ by merging the two least dissimilar classes

Inter-classes dissimilarities

Aggregation functions

Let A and B be two classes (of indices)

- single linkage (min)

$$d_S(A, B) = \min_{i \in A, j \in B} d(\mathbf{X}_i, \mathbf{X}_j)$$

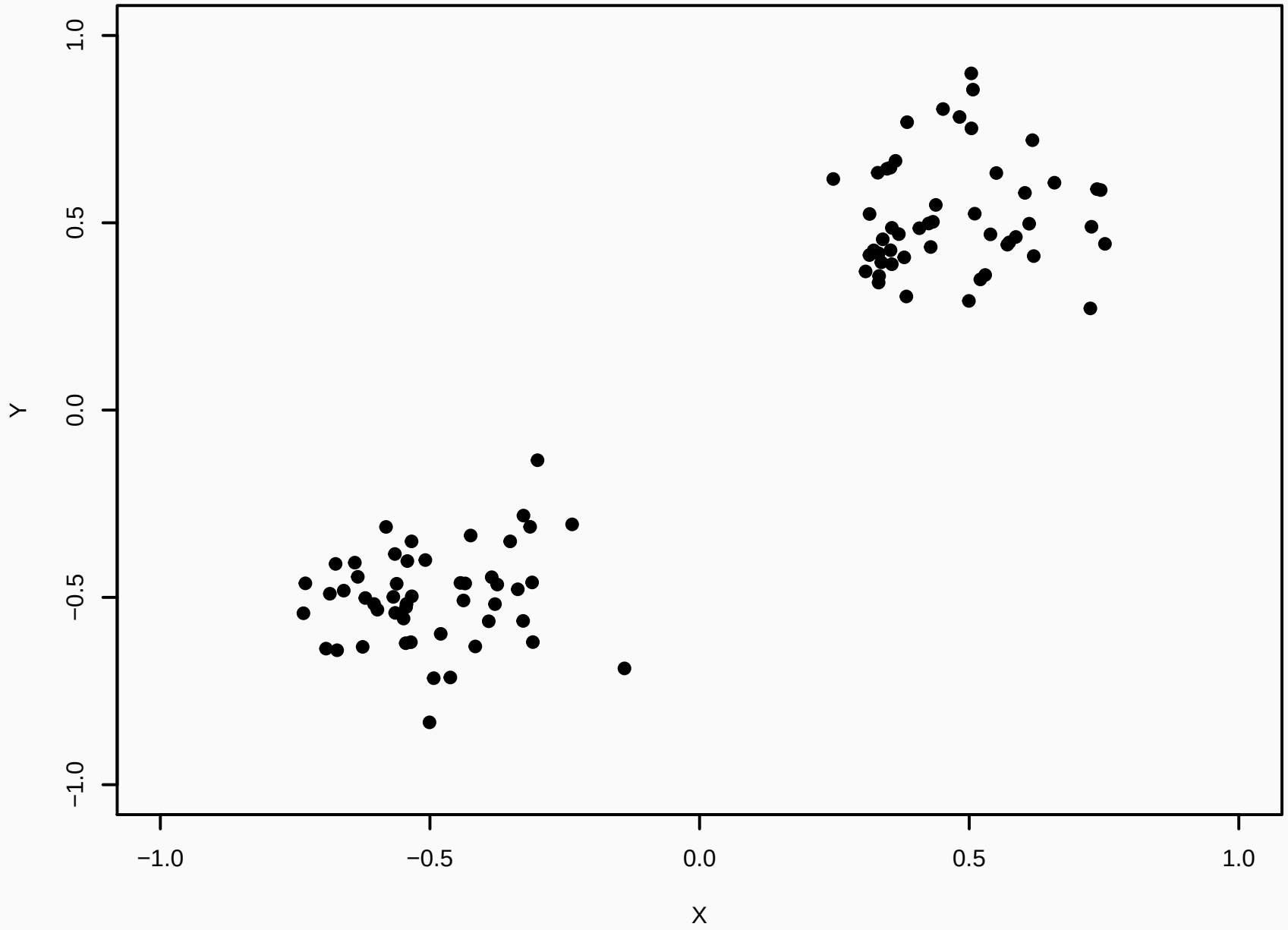
- complete linkage (max)

$$d_S(A, B) = \max_{i \in A, j \in B} d(\mathbf{X}_i, \mathbf{X}_j)$$

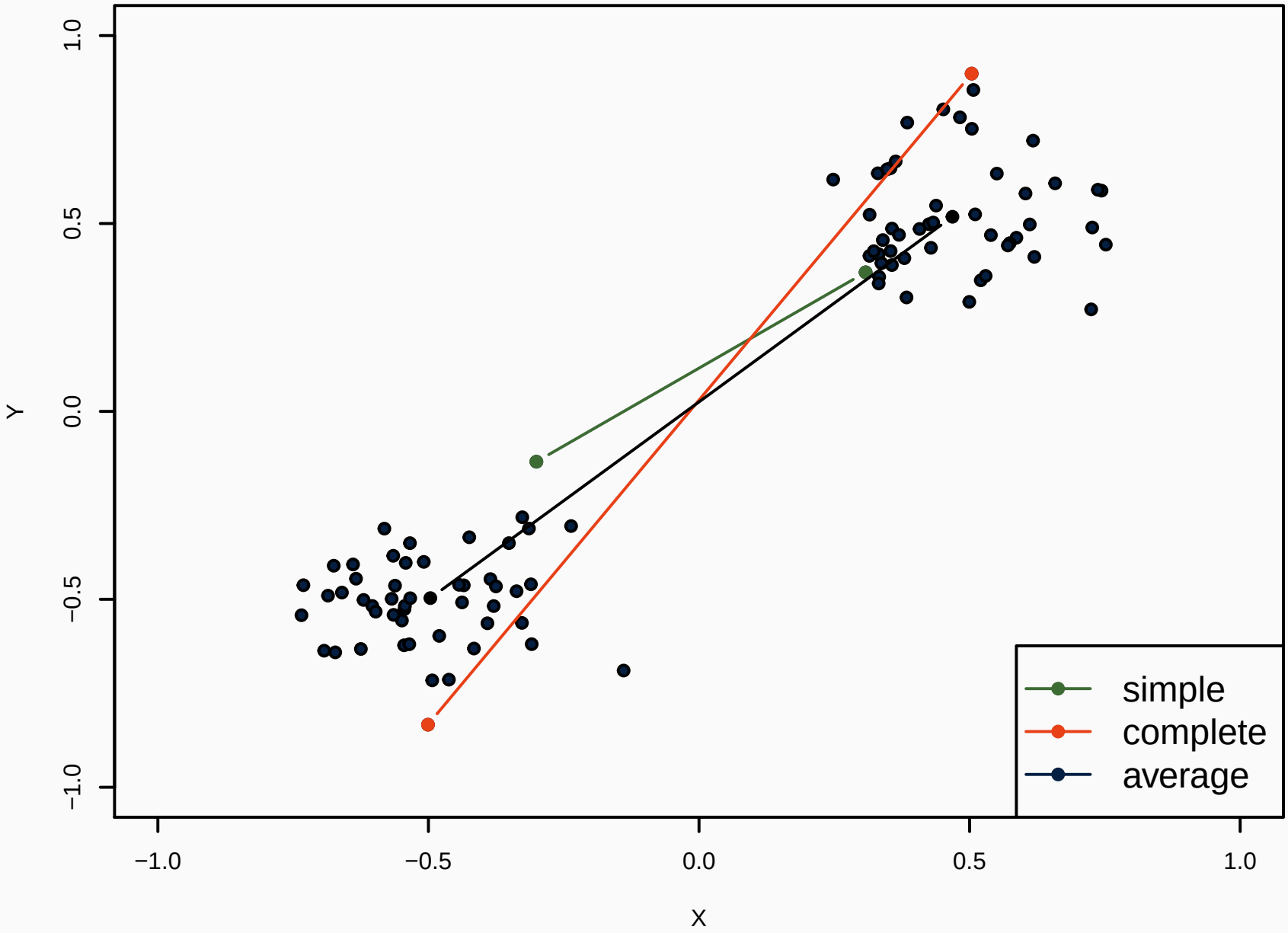
- average linkage

$$d_S(A, B) = \frac{1}{|A||B|} \sum_{i \in A, j \in B} d(\mathbf{X}_i, \mathbf{X}_j)$$

Illustration



Illustration



Computational complexity

Naive approach

- $N - 1$ steps
- step k searches for the best merge over $\frac{(N-k+1)(N-k)}{2}$ pairs of classes
- overall $\Theta\left(N^3\right)$ with adapted dissimilarity calculations
- standard implementation in many packages

Optimized solution

- simple priority list based solutions in $\Theta\left(N^2 \log N\right)$
- optimized algorithms in $\Theta\left(N^2\right)$ (see fastcluster in R)
- notice that the dissimilarity must be calculated! For $\mathcal{X} = \mathbb{R}^P$, this adds in general a $\Theta\left(N^2 P\right)$ cost.

In practice

Outline

1. choose a dissimilarity
2. choose an aggregation function
3. build the hierarchy
4. study the dendrogram:
   - the heights are the dissimilarities between merged classes
   - gaps can be seen as abrupt changes in merge qualities
   - visualization methods can be used to display the classes (e.g. Principal Component Analysis)

In practice

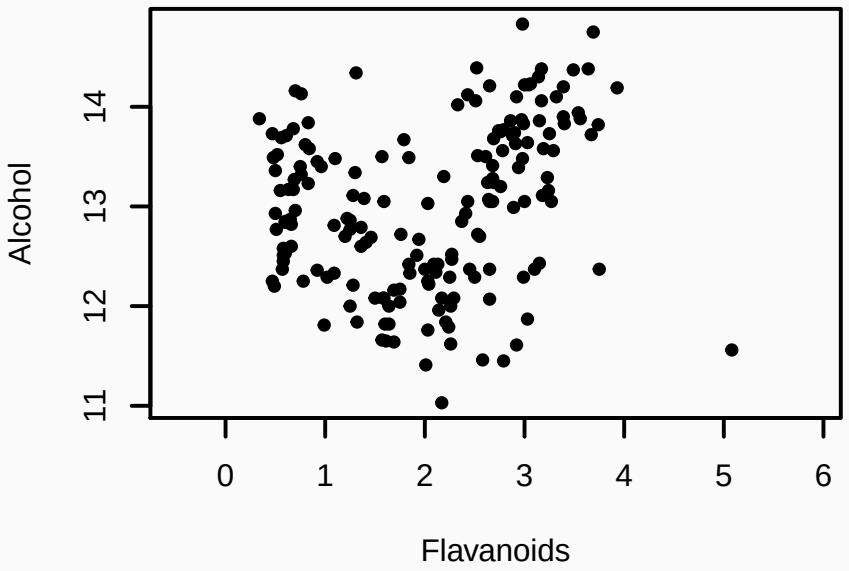
Outline

1. choose a dissimilarity
2. choose an aggregation function
3. build the hierarchy
4. study the dendrogram:
   - the heights are the dissimilarities between merged classes
   - gaps can be seen as abrupt changes in merge qualities
   - visualization methods can be used to display the classes (e.g. Principal Component Analysis)

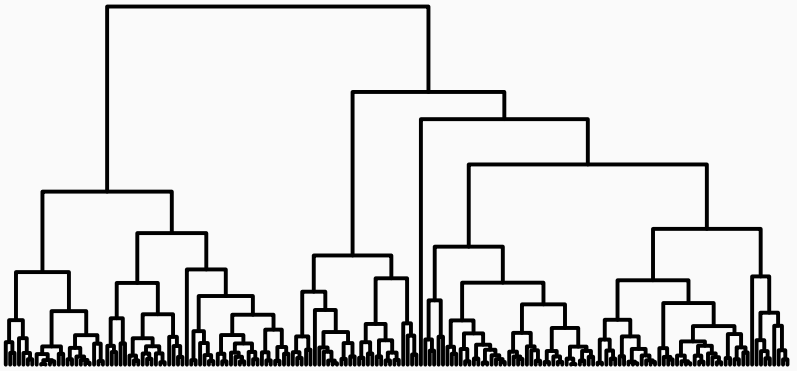
Limitations

- results strongly depend on both the dissimilarity and the aggregation function
- somewhat slow
- exploratory tool rather than clustering tool

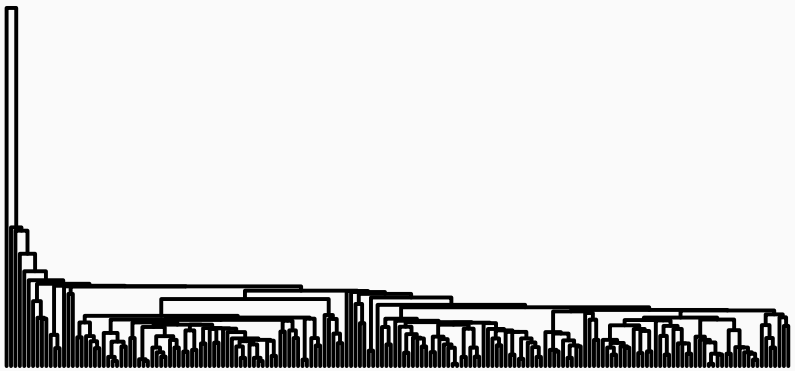
Example



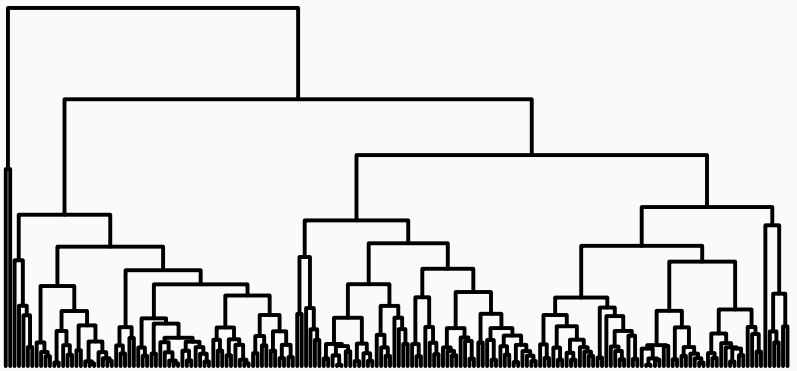
Complete



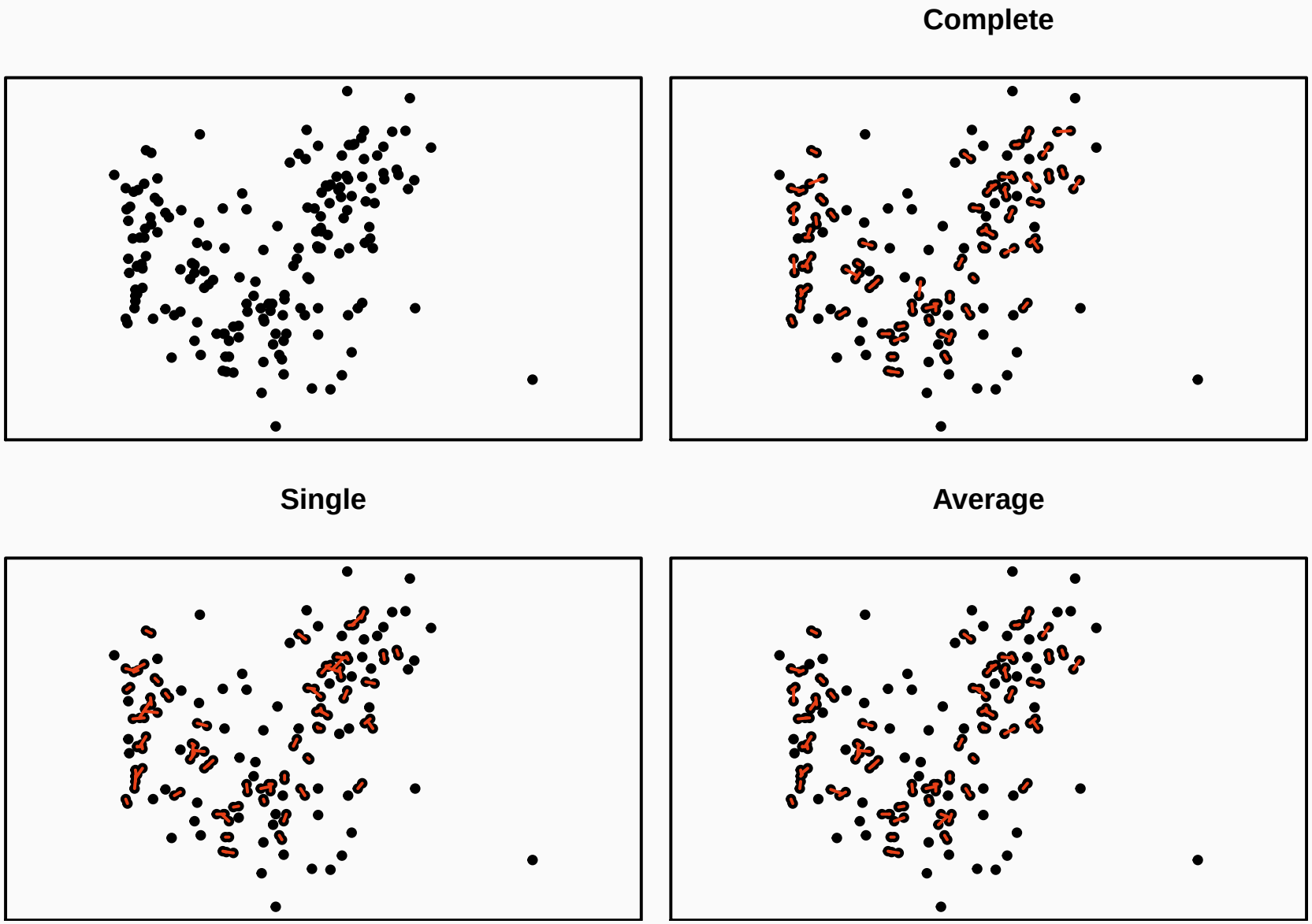
Single



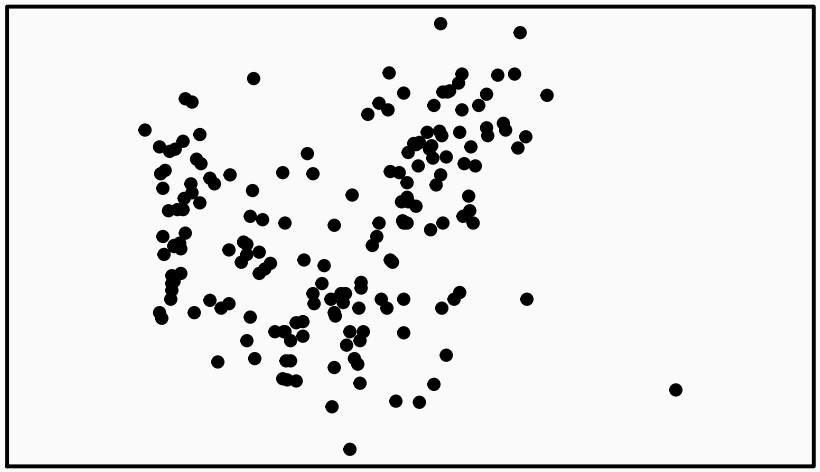
Average



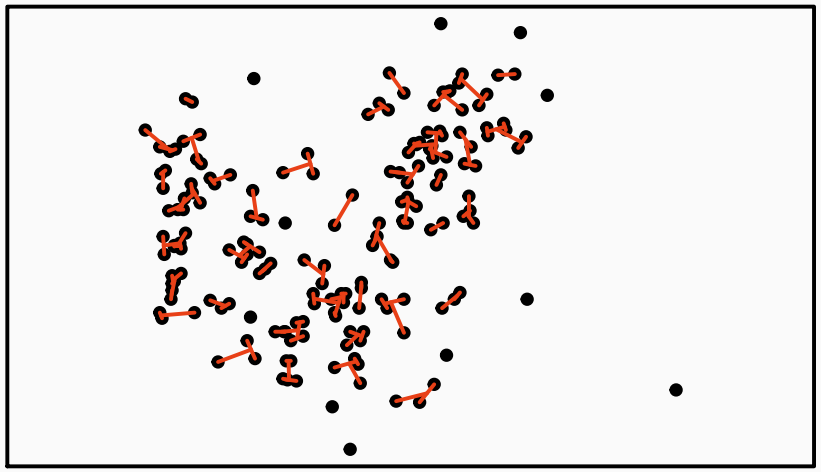
Example



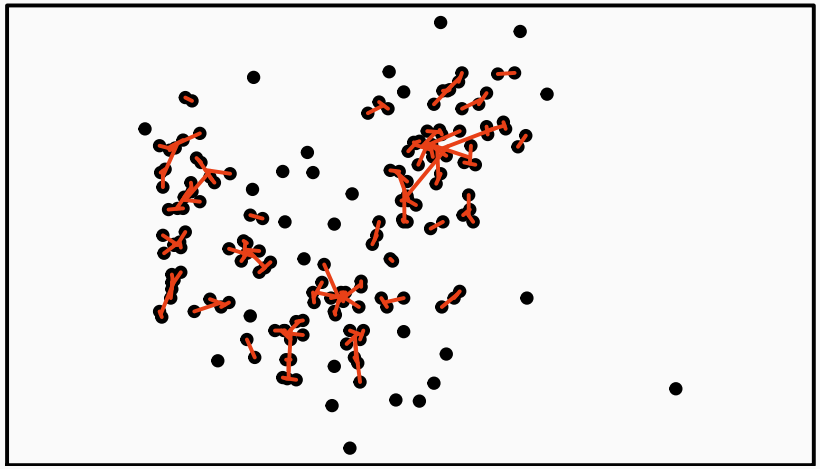
Example



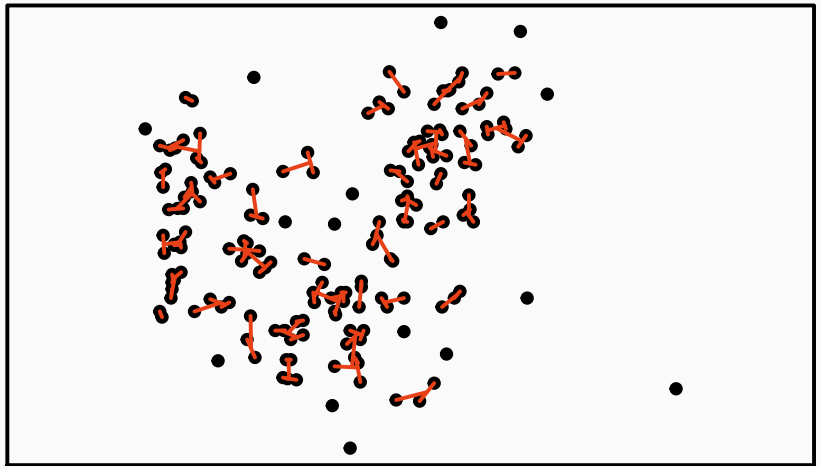
Complete



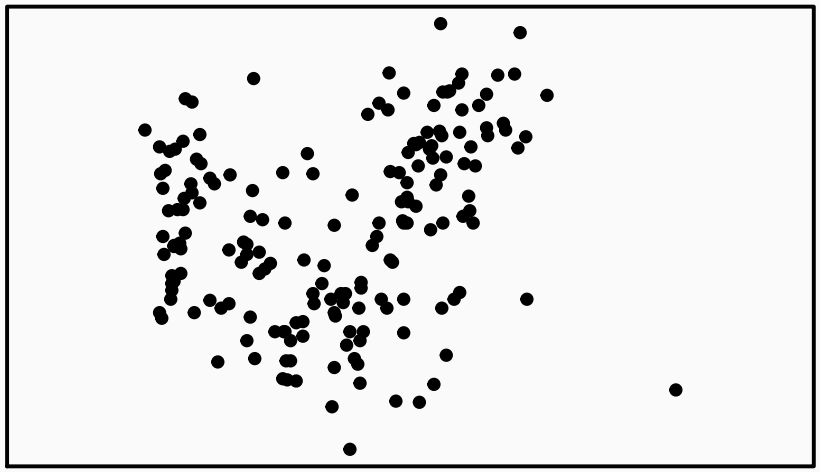
Single



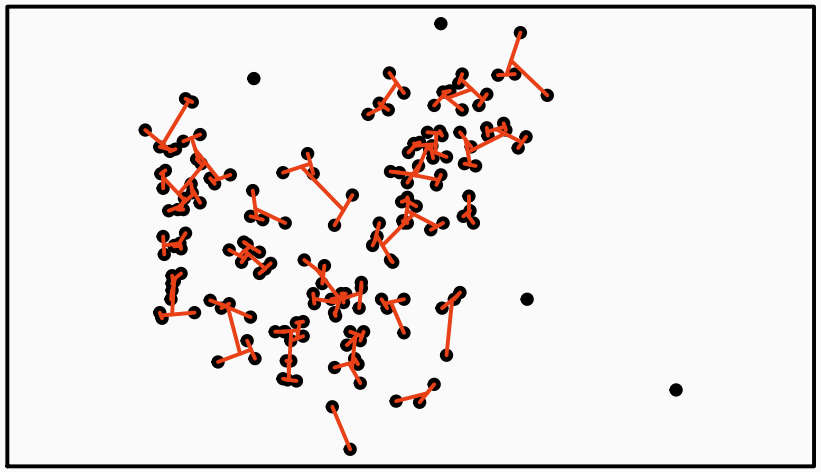
Average



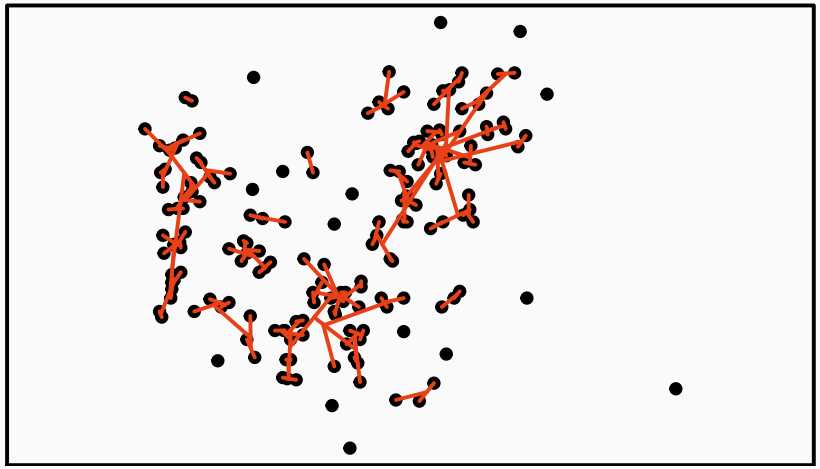
Example



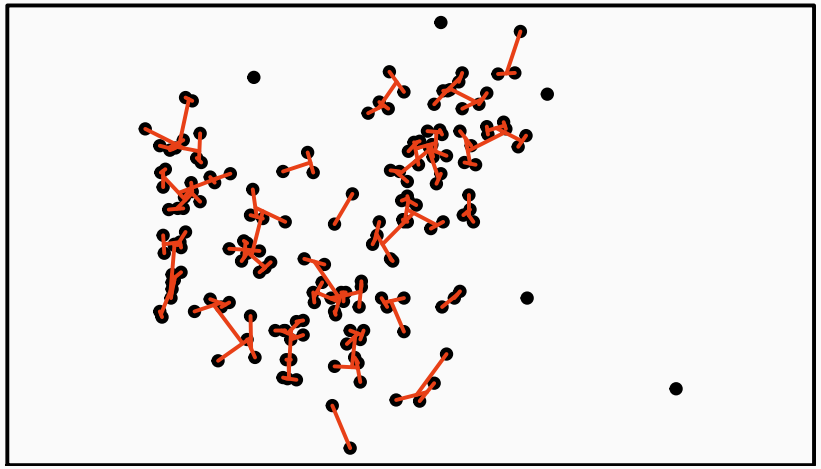
Complete



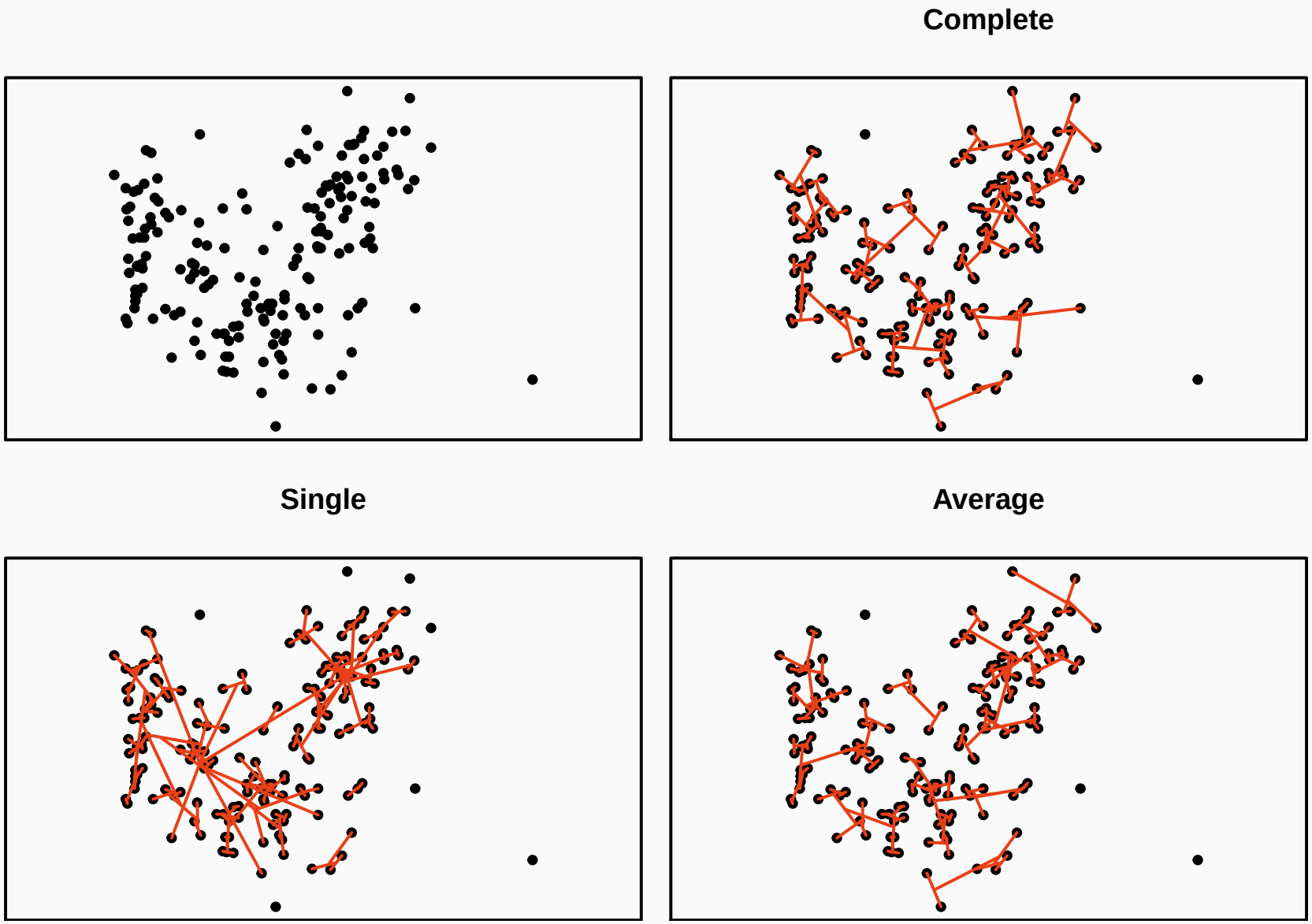
Single



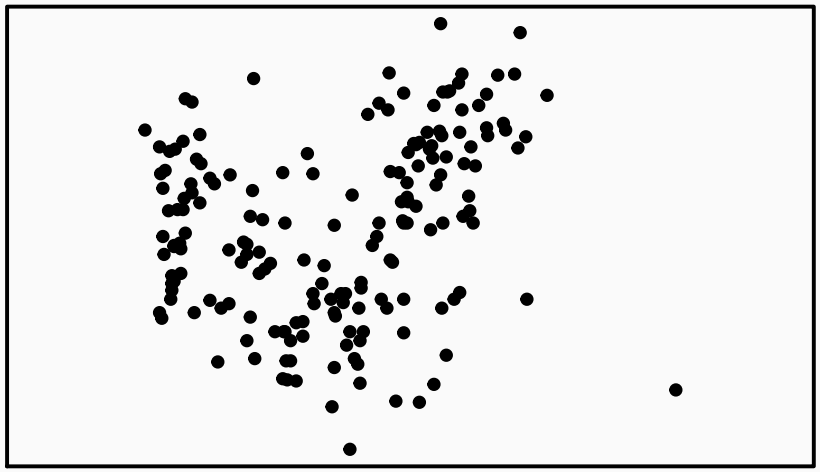
Average



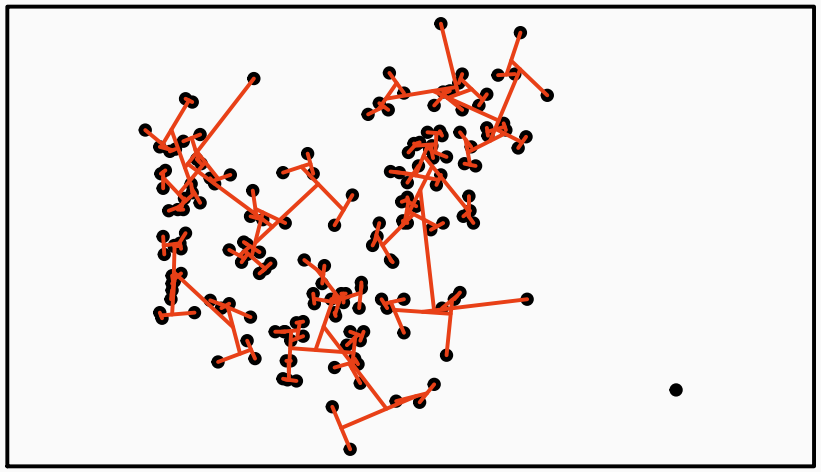
Example



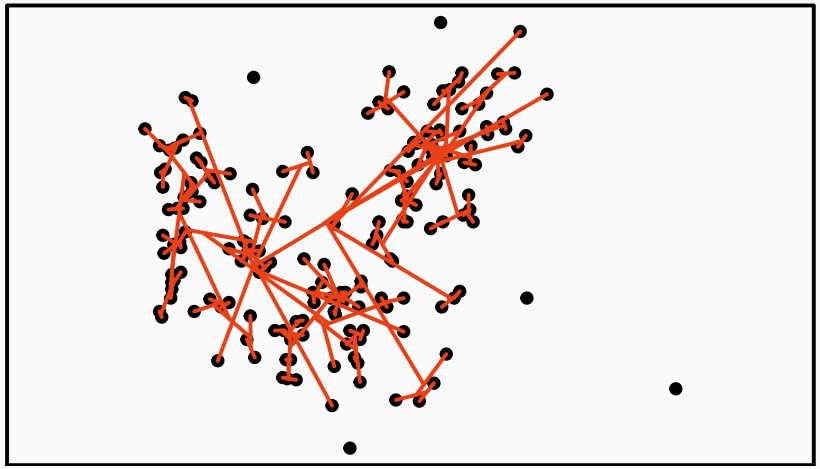
Example



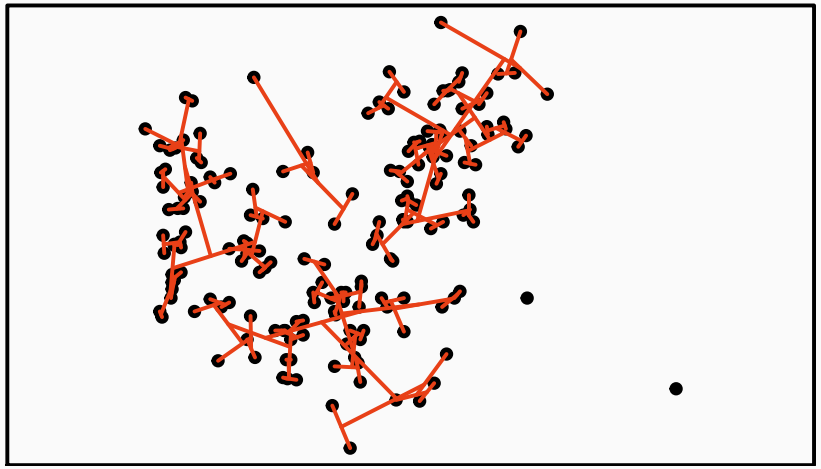
Complete



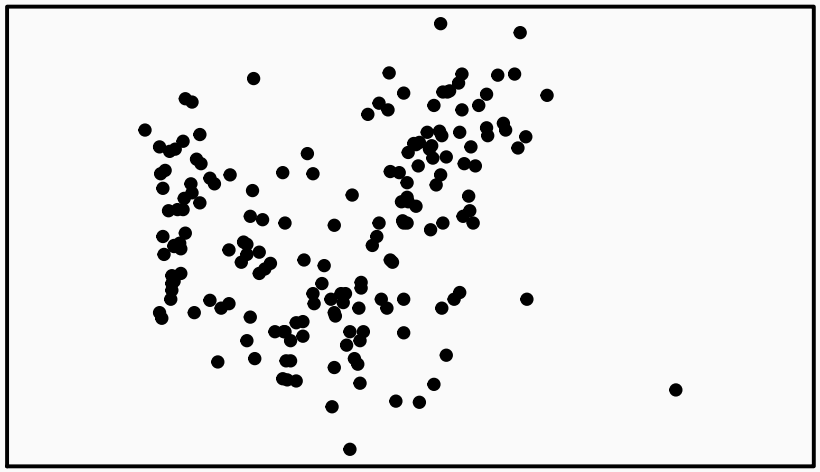
Single



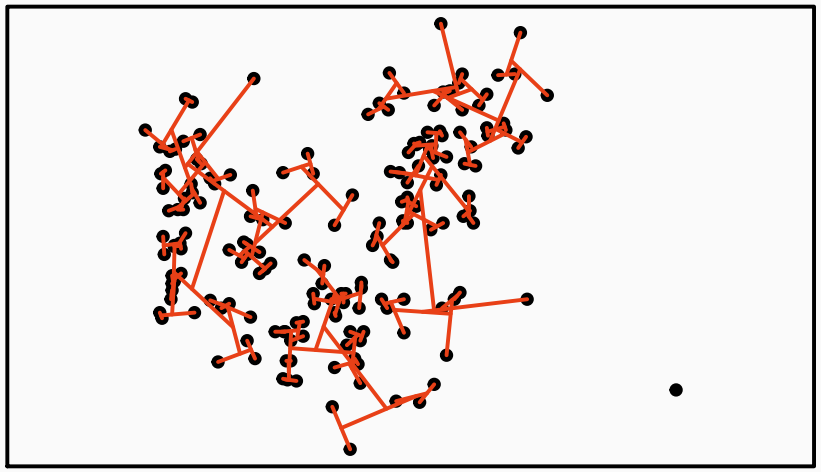
Average



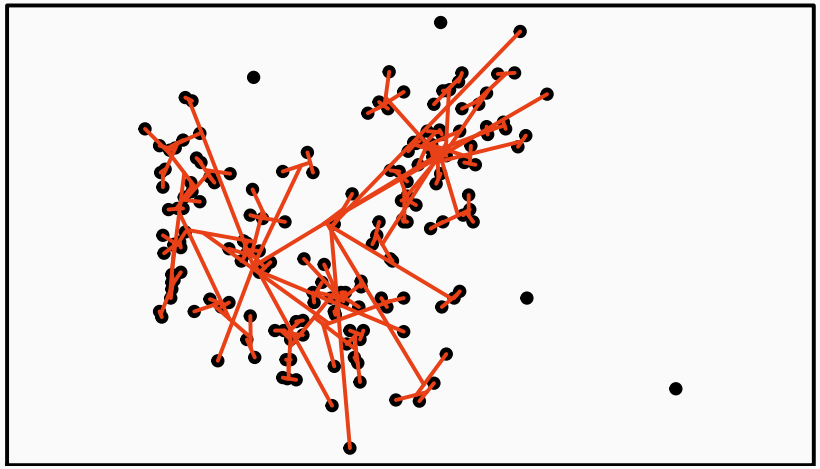
Example



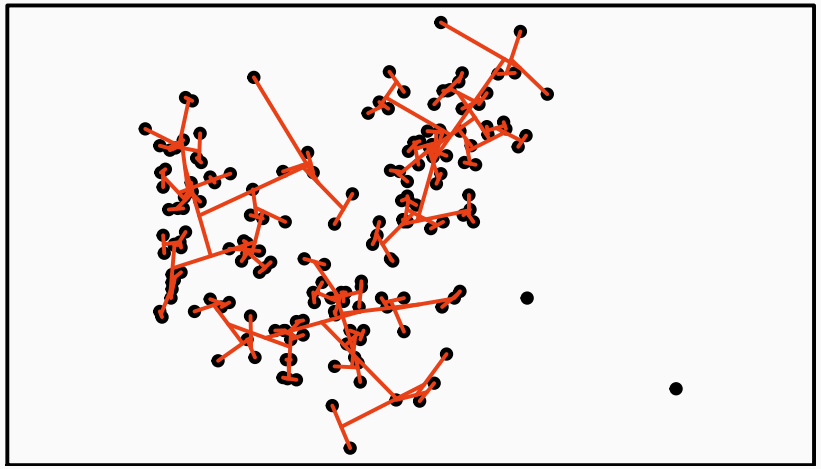
Complete



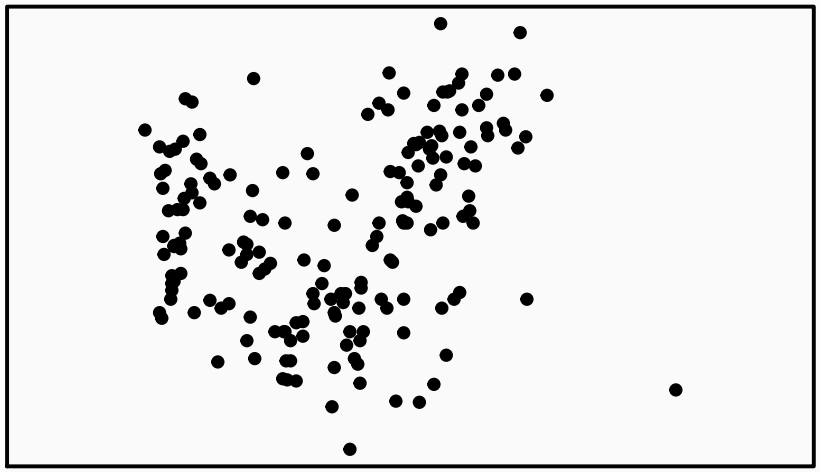
Single



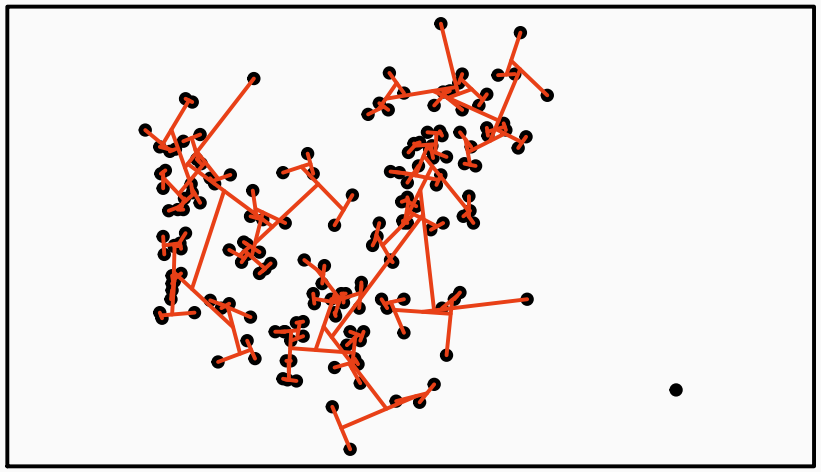
Average



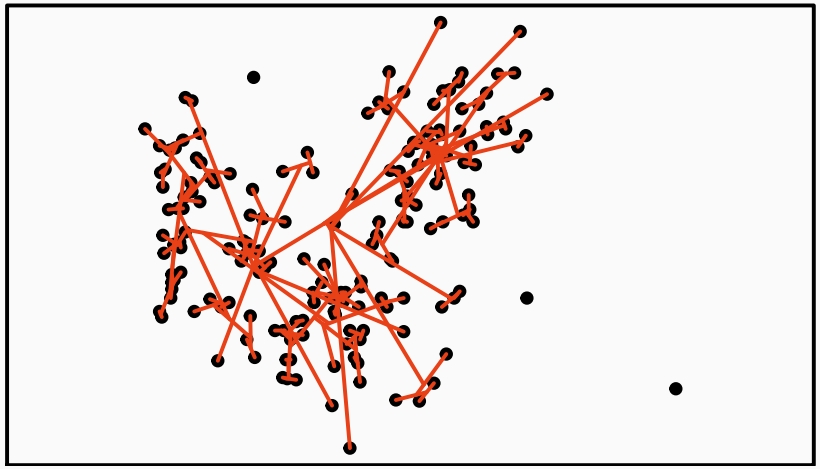
Example



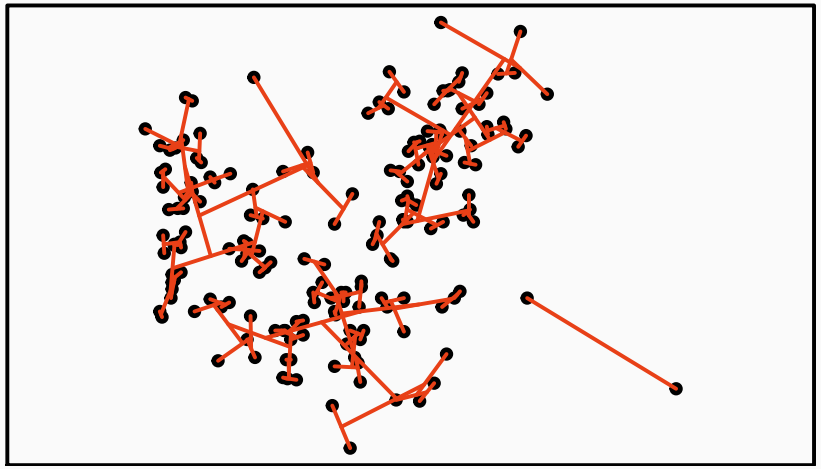
Complete



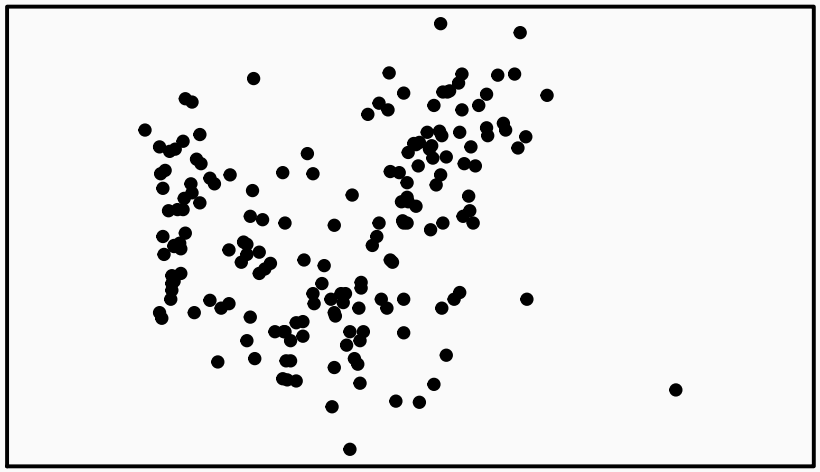
Single



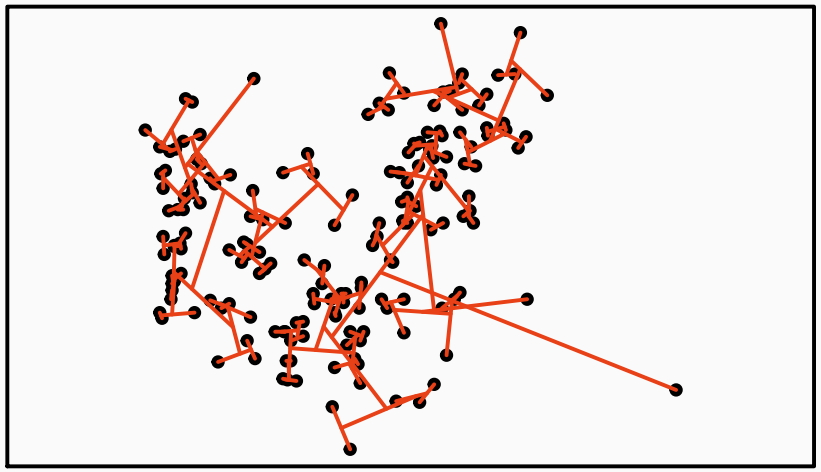
Average



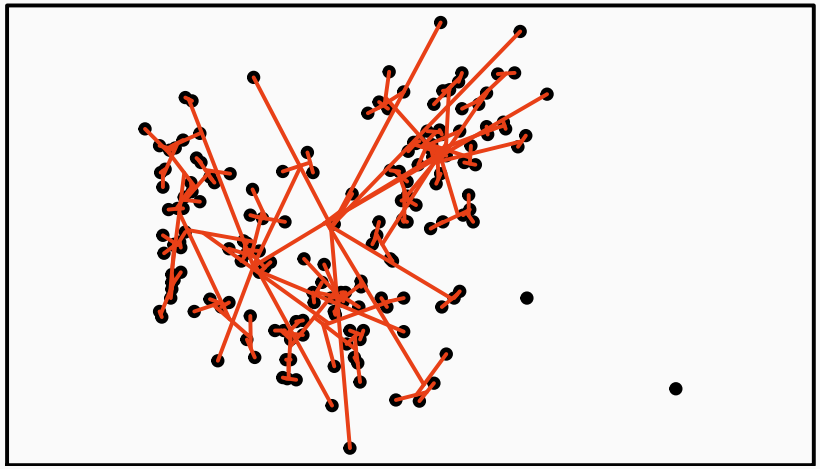
Example



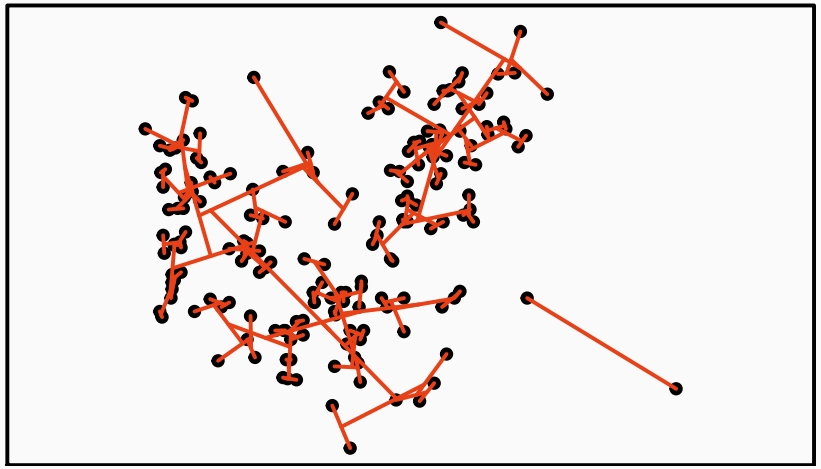
Complete



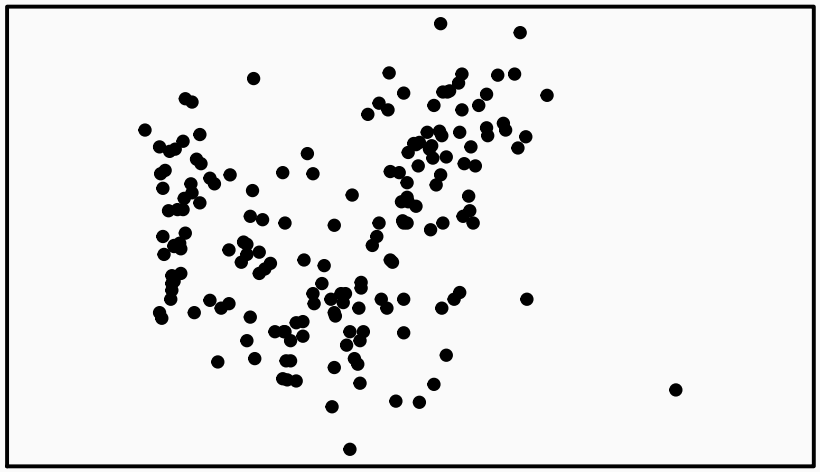
Single



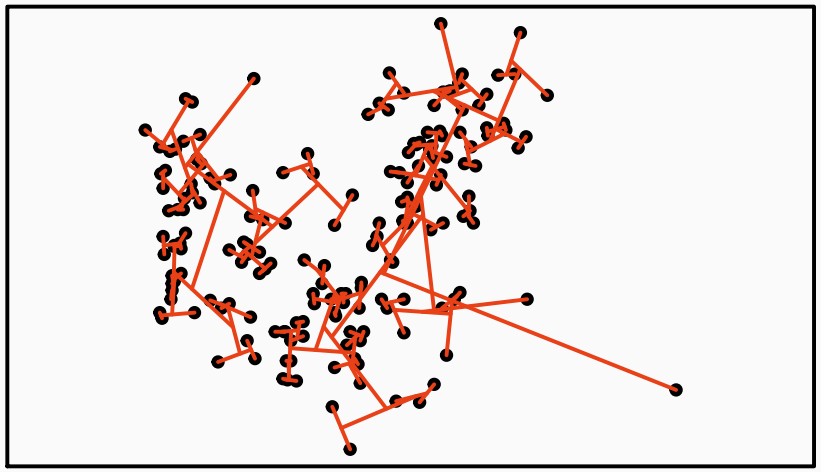
Average



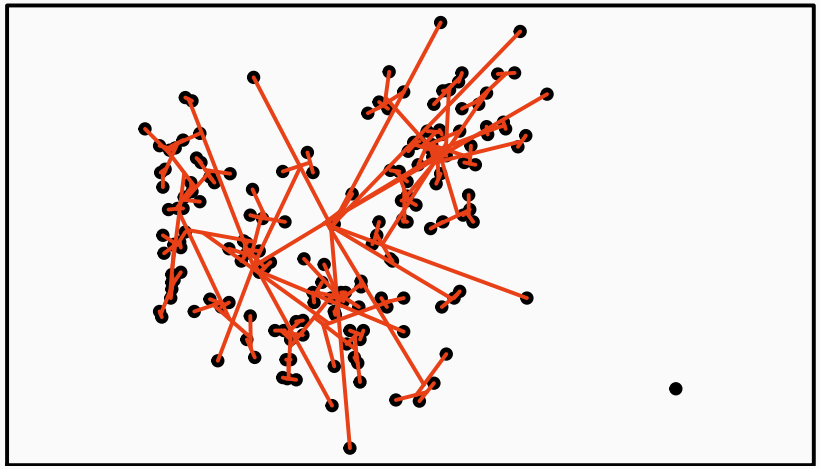
Example



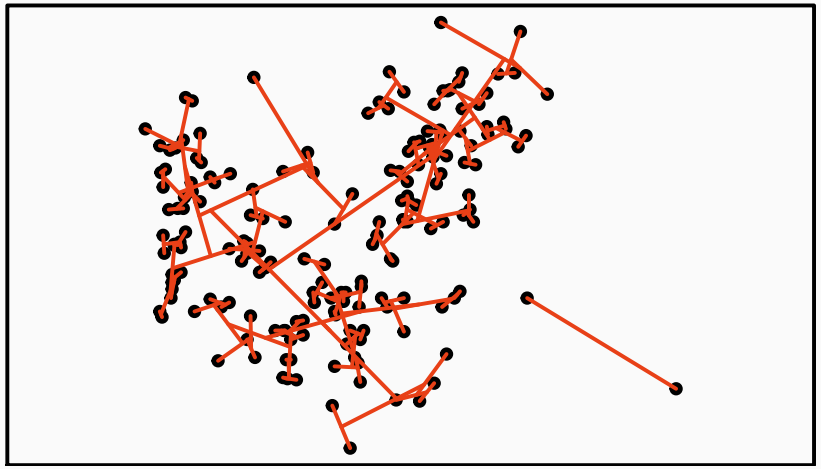
Complete



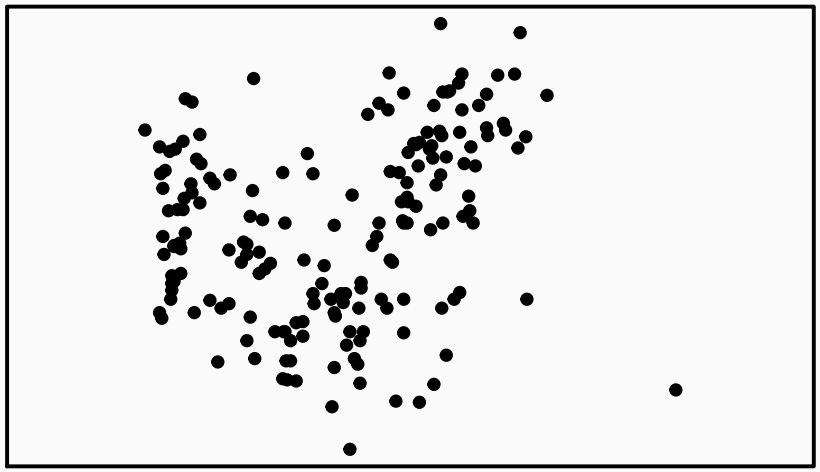
Single



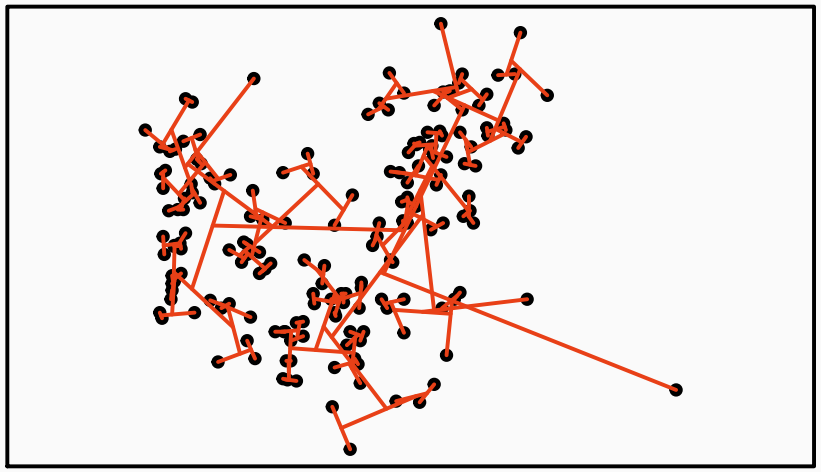
Average



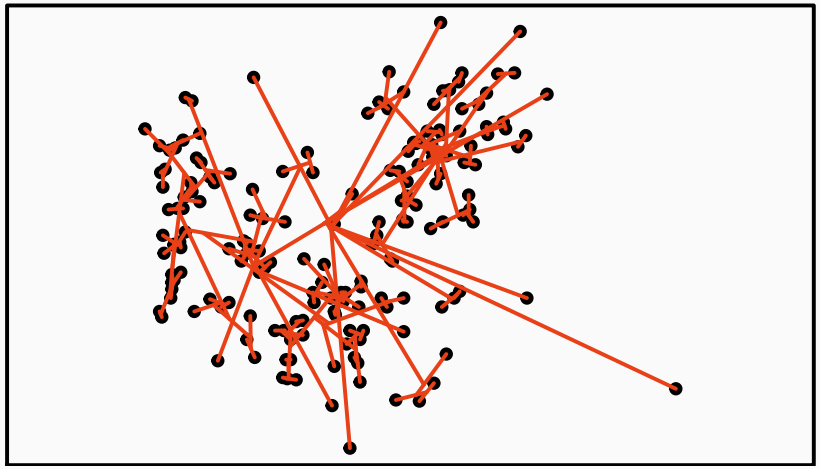
Example



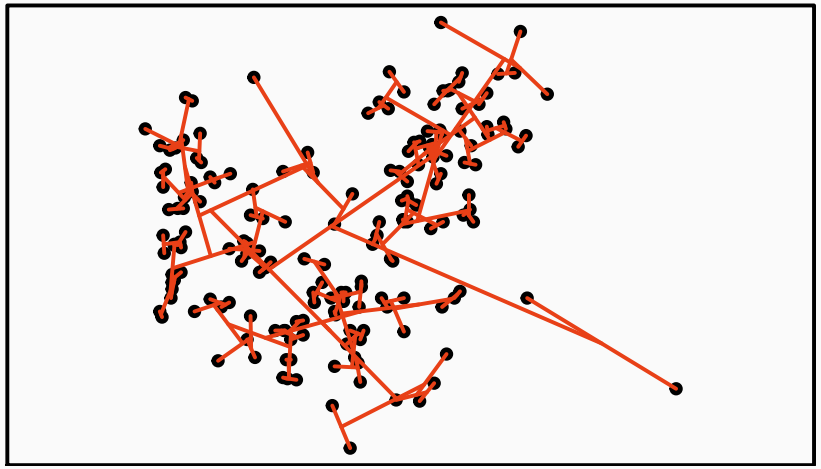
Complete



Single



Average



Example

Remarks

- initial behaviors are quite independent from the aggregation function
- single linkage:
  - tendency to produce “long” classes (chaining)
  - outlier isolation
- complete linkage:
  - balanced class sizes
  - classes can be close one to another (crowding)
- average linkage
  - average behavior...
  - resistant to noise but sensitive to dissimilarity transformation

Cluster quality

Classes as clusters

A partition provides a good *clustering* if its classes are

- homogeneous
- well separated

Criteria

- homogeneity via the diameter

$$D(A) = \max_{i \in A, j \in A} d(\mathbf{X}_i, \mathbf{X}_j)$$

- separation via dissimilarity
- a good clustering: small diameters and large dissimilarities!

Aggregation function

Single linkage

- no control over $D(A)$ during the merge
- could end up with classes with very different diameters
- distances between classes can be very small relatively to their diameter

Complete linkage

- the diameter is the quality measure of a merge!
- however, no control at all over separability

Two extreme cases

with the average linkage in between...

In practice

General rules of thumb

- single linkage
  - gives frequently poor results
  - might be useful to spot outliers
- average linkage
  - gives generally good clusters
  - interesting compromise between diameter and separation
- complete linkage
  - maximally homogeneous classes
  - useful when there are no real clusters (i.e., clusters that can be easily separated)

Partition quality

Quality versus distance

- aggregation functions express the quality of a merge in terms of distances (min, max or average)
- alternative solution: express the quality using the resulting cluster

Within "variance"

- within variance of a class A (homogeneity measure)

$$W(A) = \frac{1}{|A|} \sum_{i \in A, j \in A} d(\mathbf{X}_i, \mathbf{X}_j)$$

- total within variance of a partition $\mathcal{P} = \{A_1, \ldots, A_K\}$

$$W(\mathcal{P}) = \sum_{k=1}^{K} W(A_k)$$

Ward's method

Optimizing the within variance

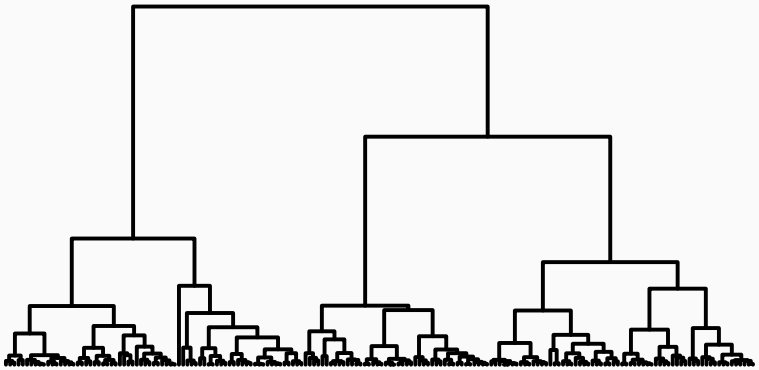
- same aggregative algorithm than with dissimilarities
- (non)quality of a merge:
  - the increase in within variance induced by merging A with B
  - local computation

$$\Delta_{(A,B)\to A\cup B} = W(A \cup B) - W(A) - W(B)$$

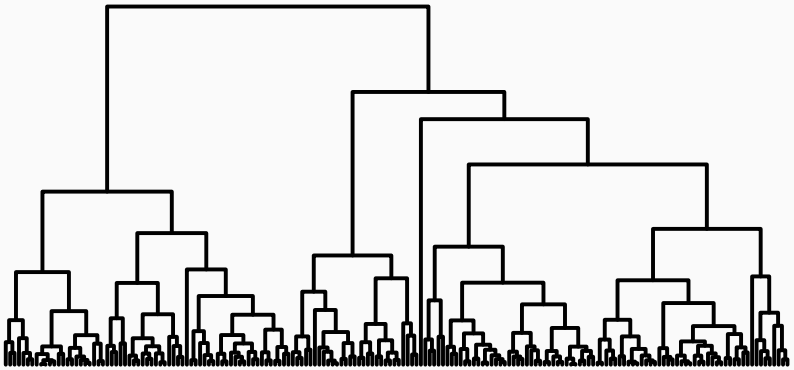
- can be seen as a greedy optimization of $W(\mathcal{P})$ at each stage of the hierarchy

Example

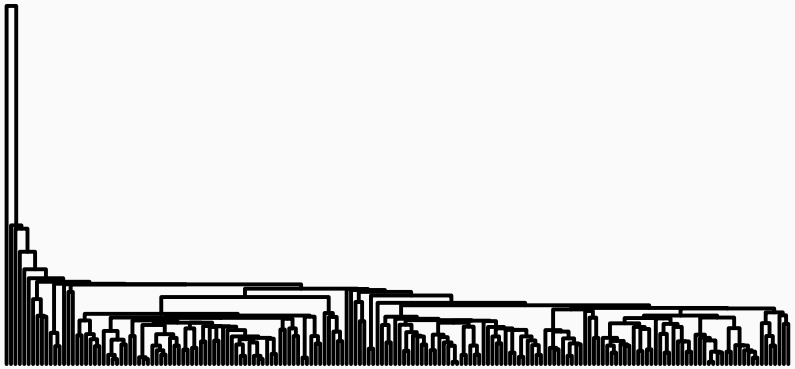
Ward



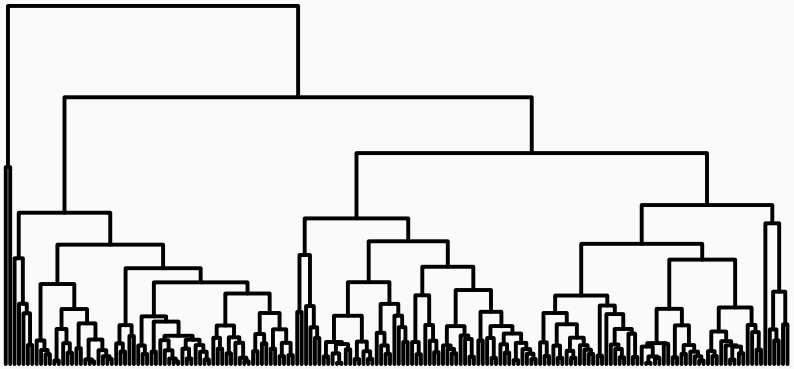
Complete



Single

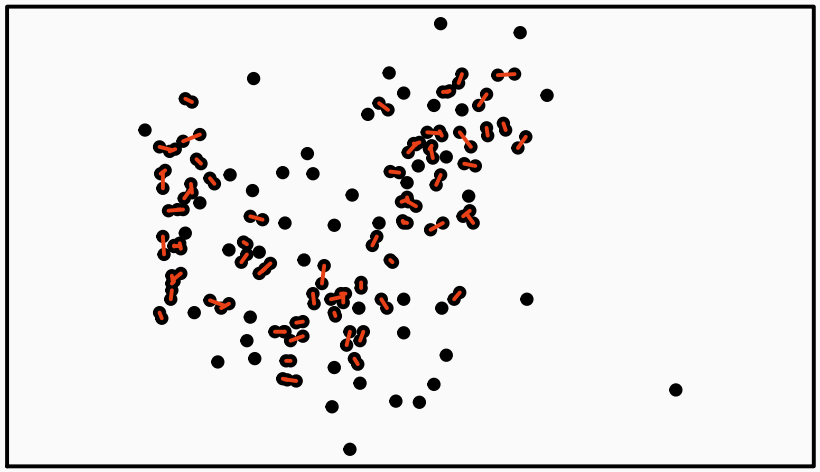


Average

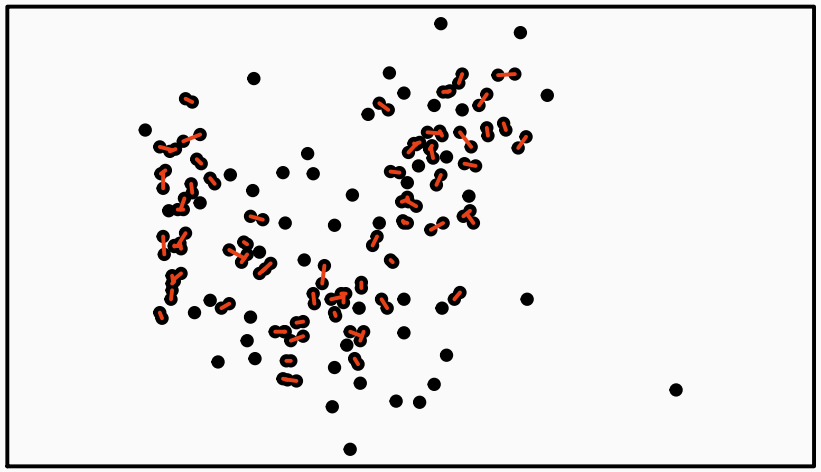


Example

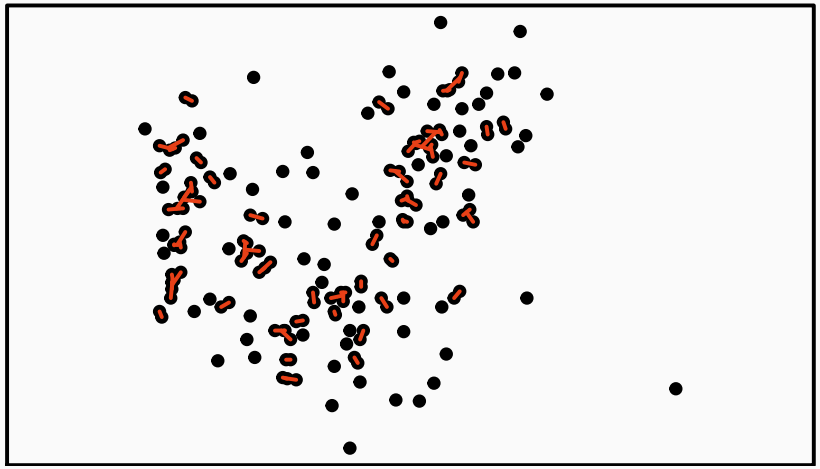
Ward



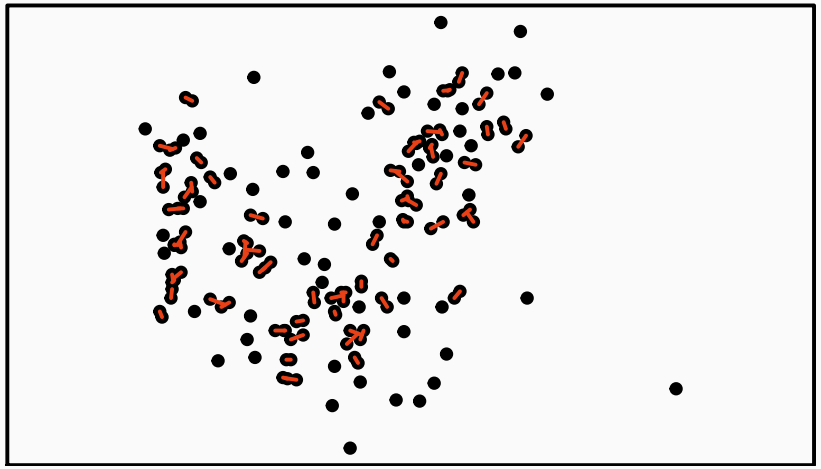
Complete



Single

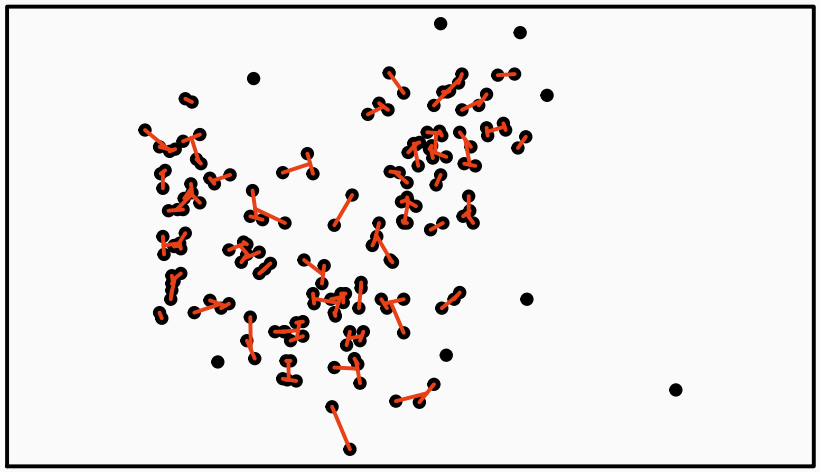


Average

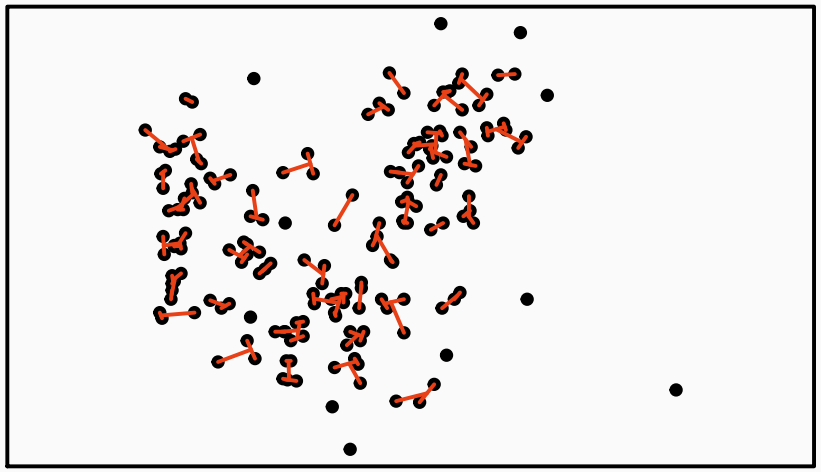


Example

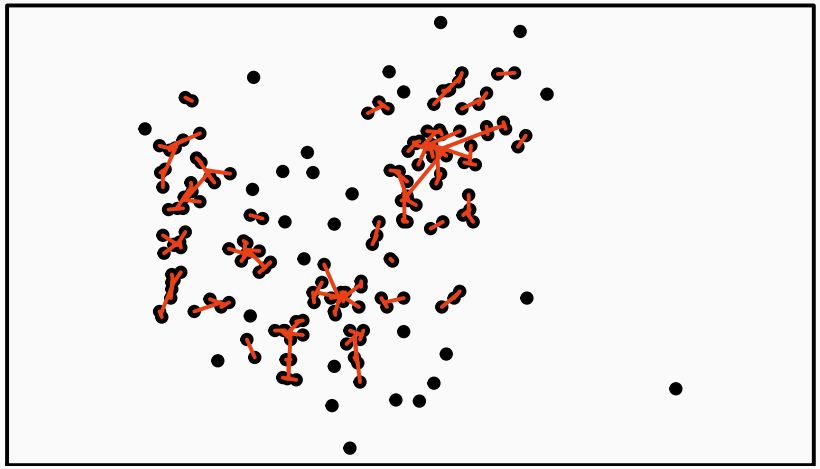
Ward



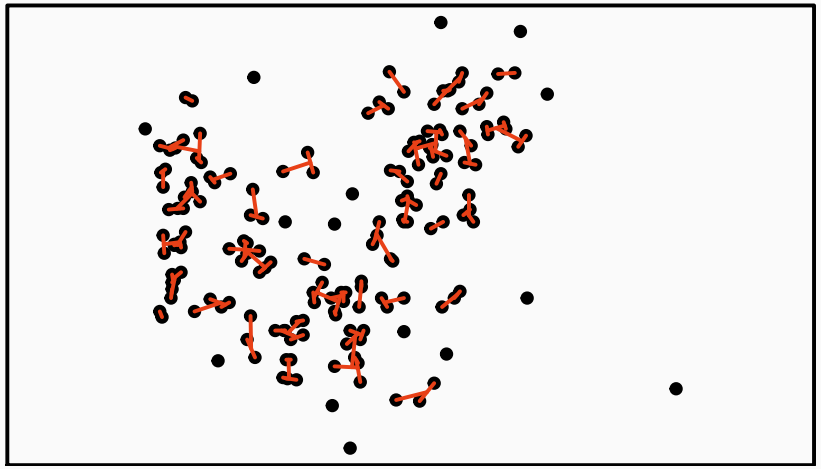
Complete



Single

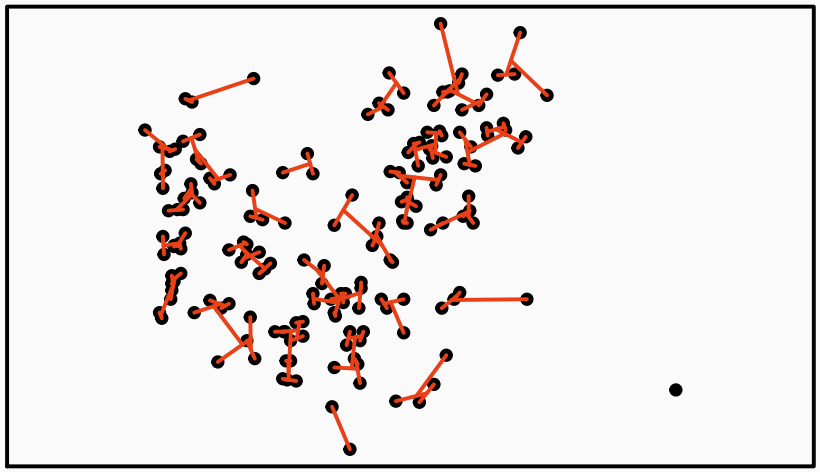


Average

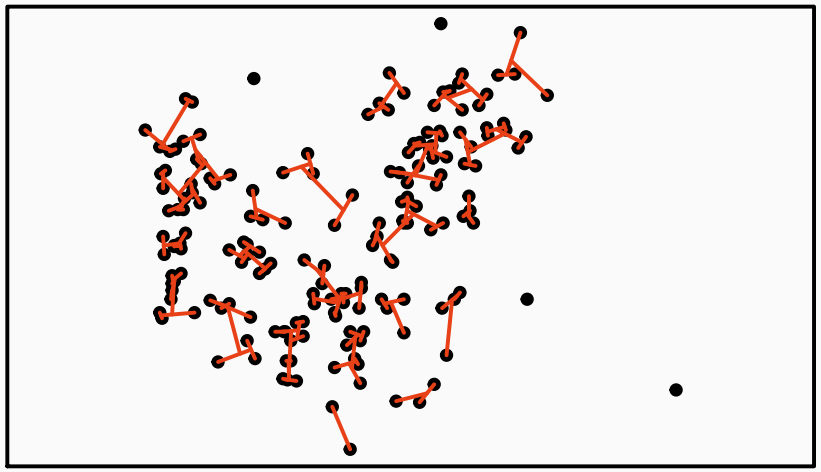


Example

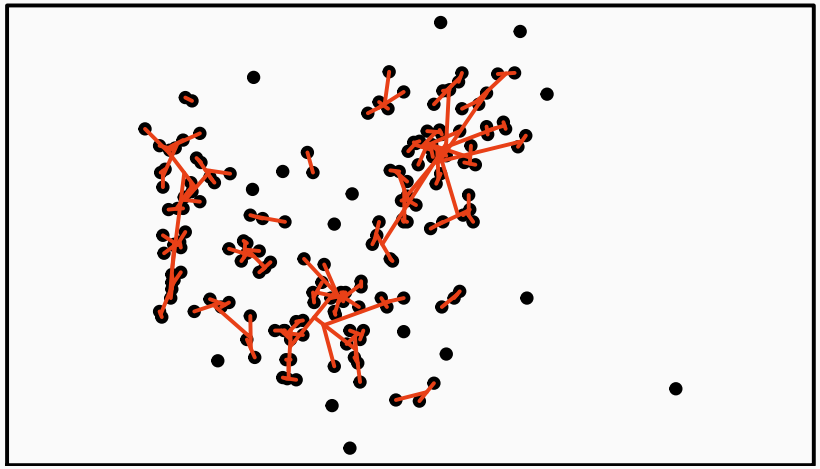
Ward



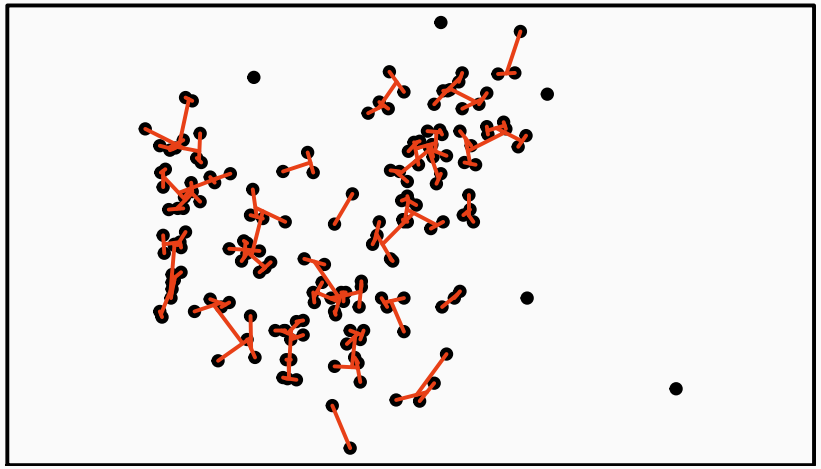
Complete



Single

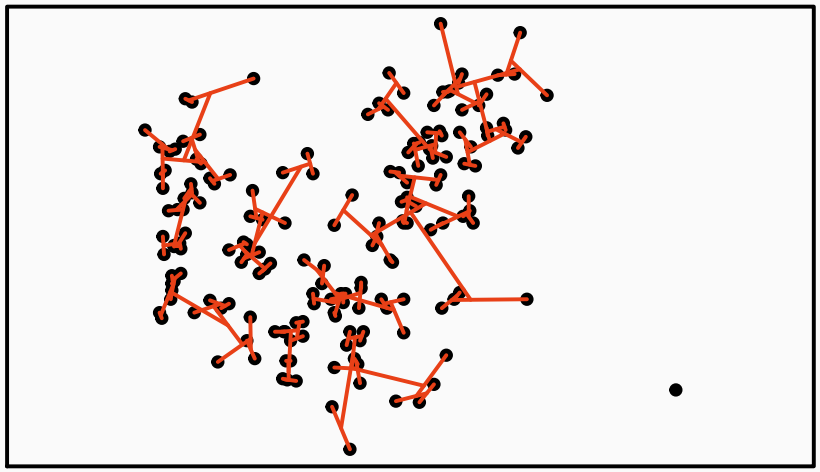


Average

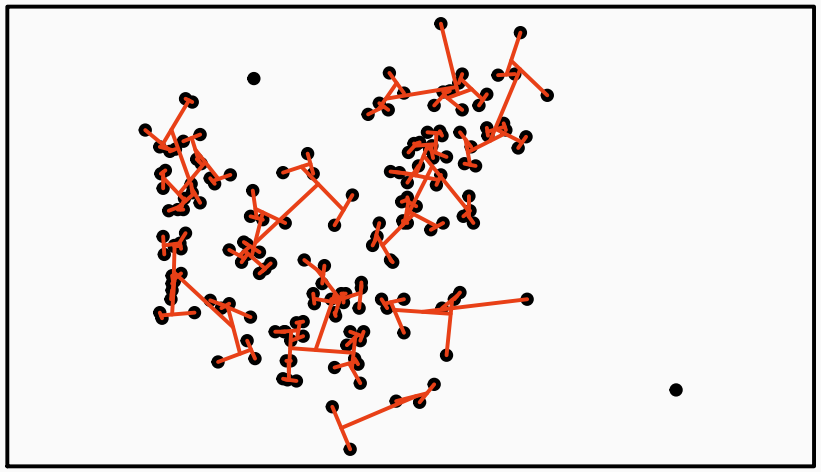


Example

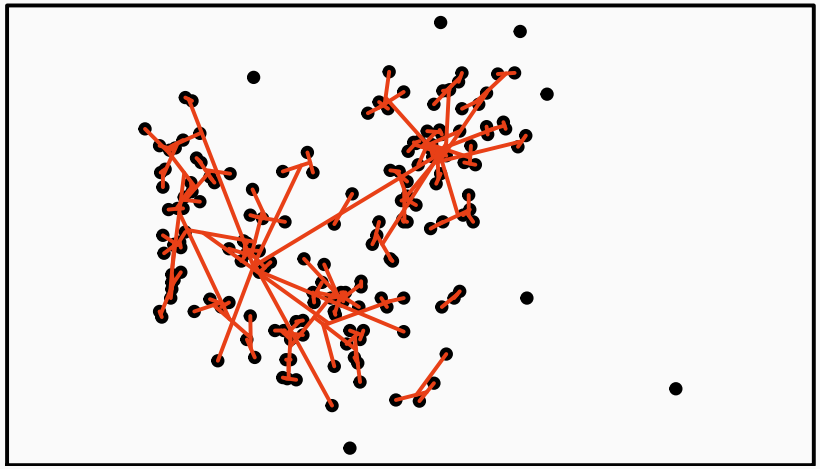
Ward



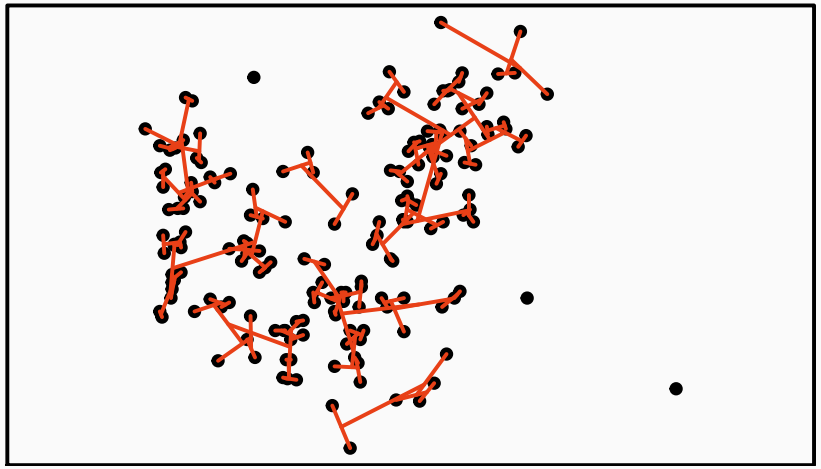
Complete



Single

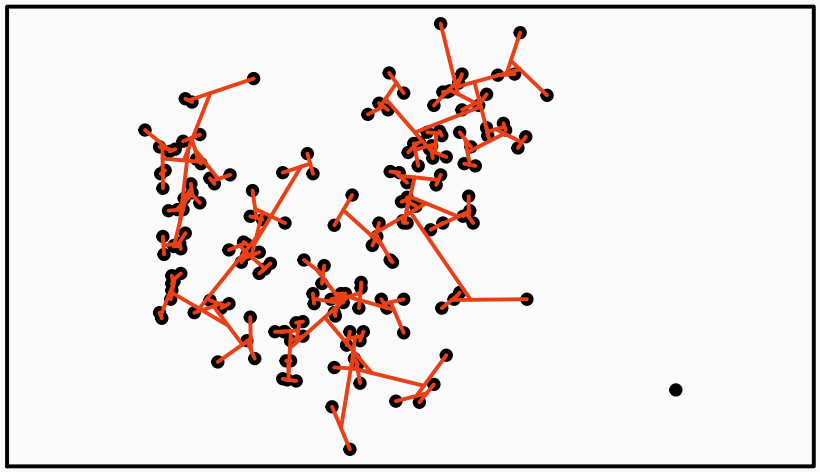


Average

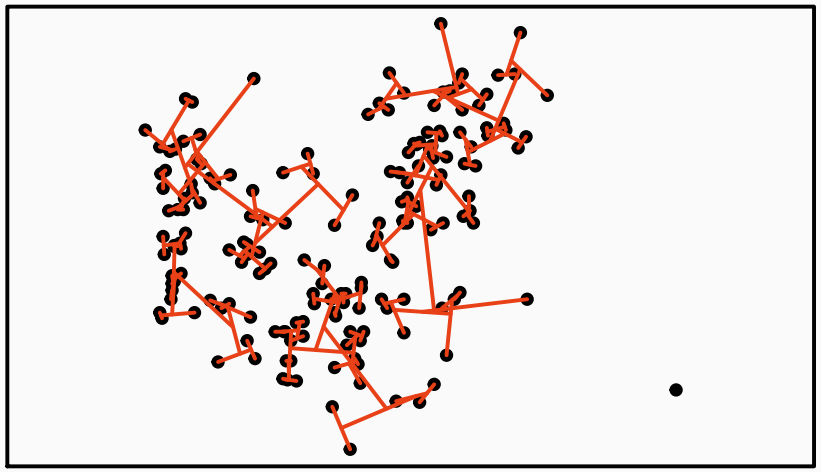


Example

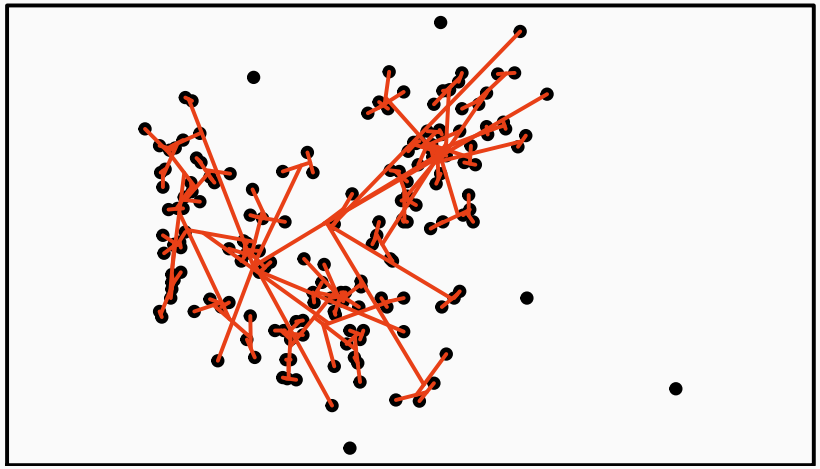
Ward



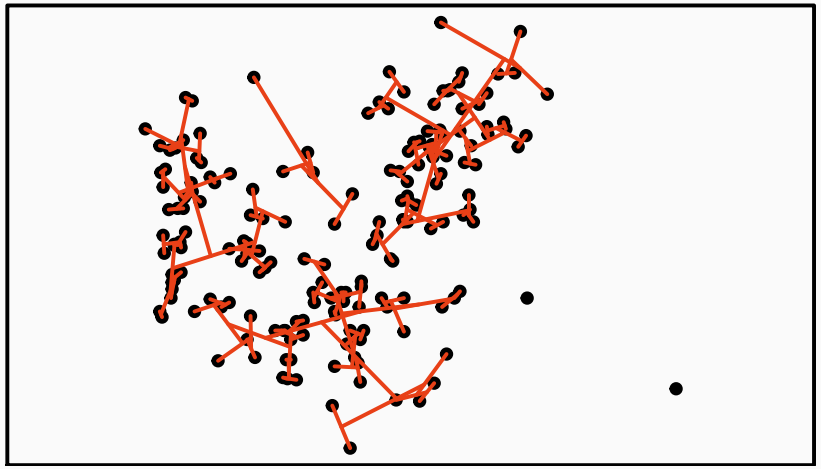
Complete



Single

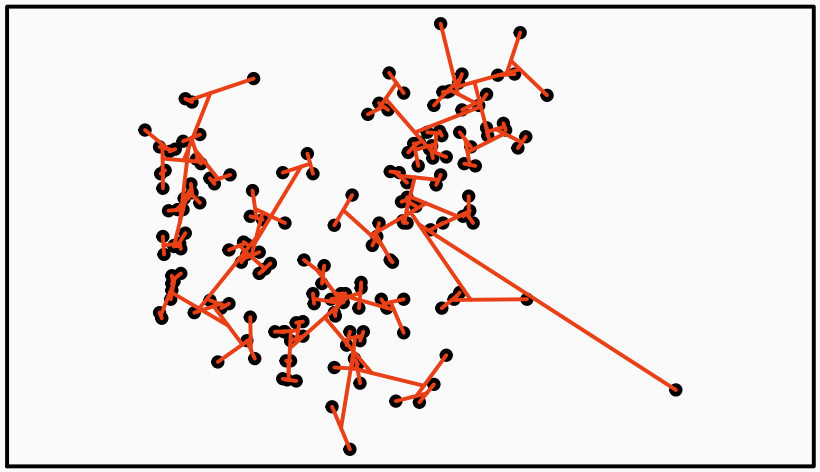


Average

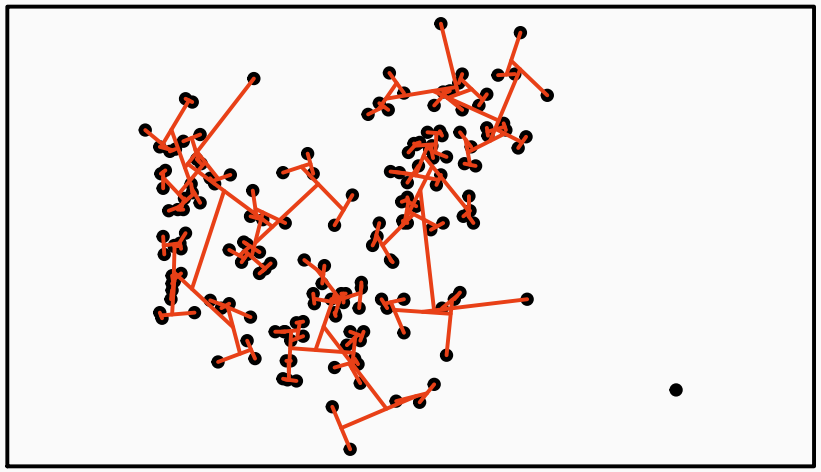


Example

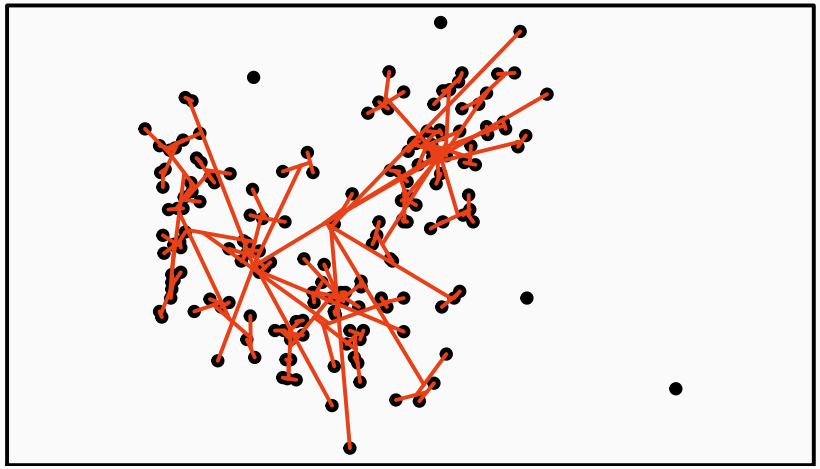
Ward



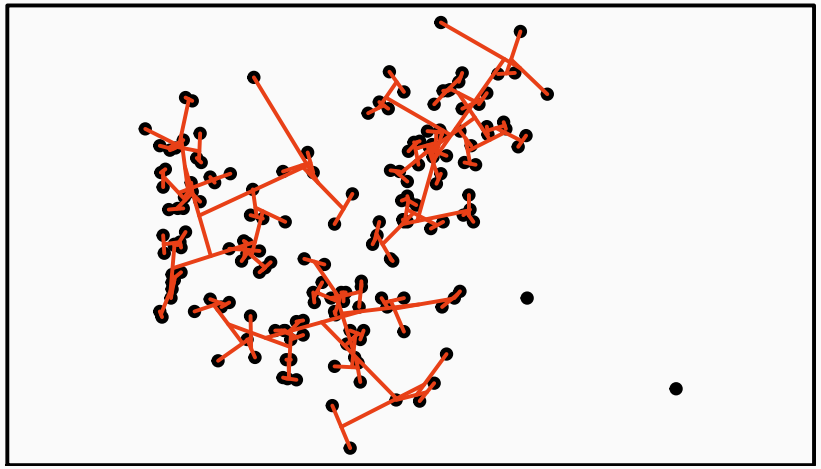
Complete



Single

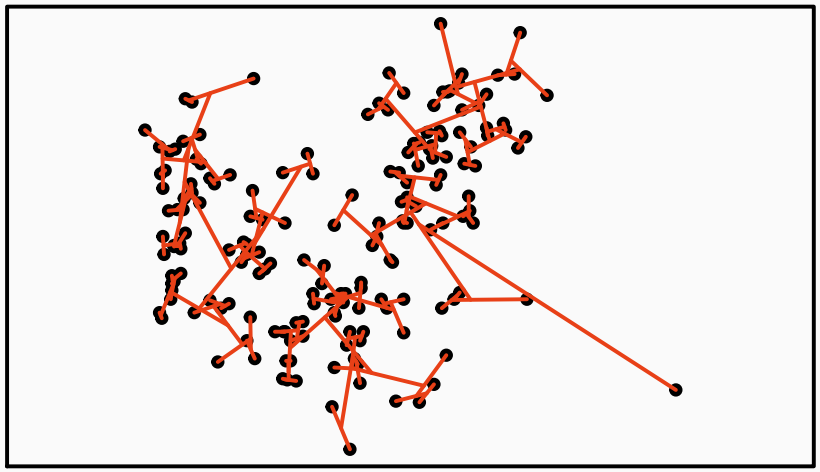


Average

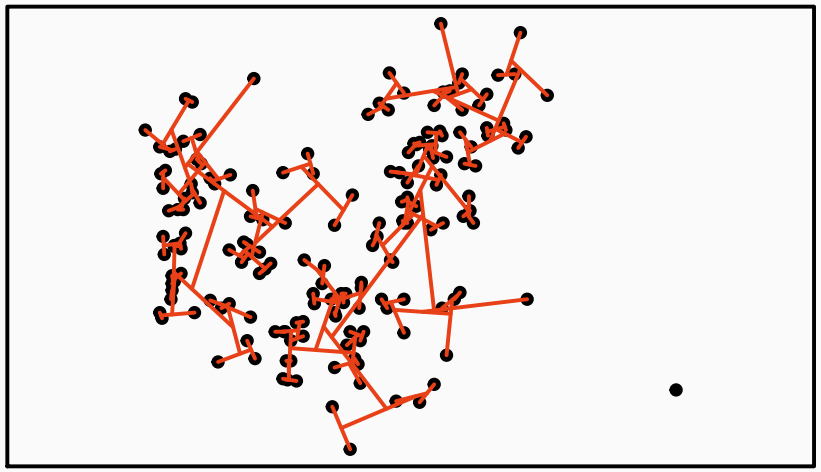


Example

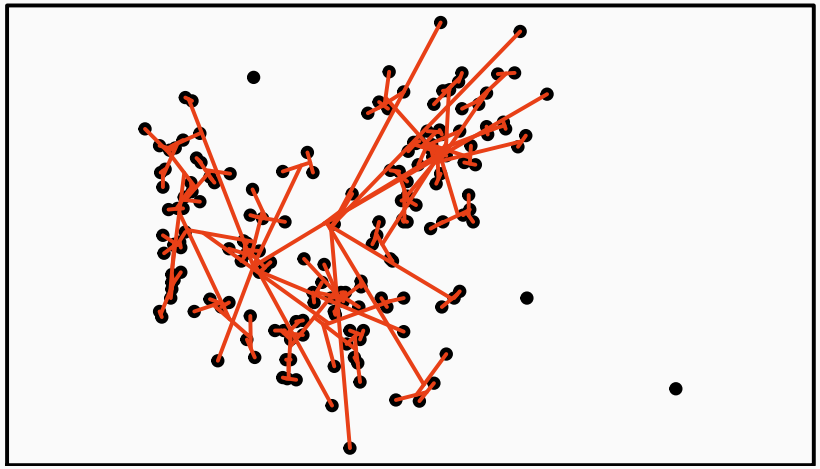
Ward



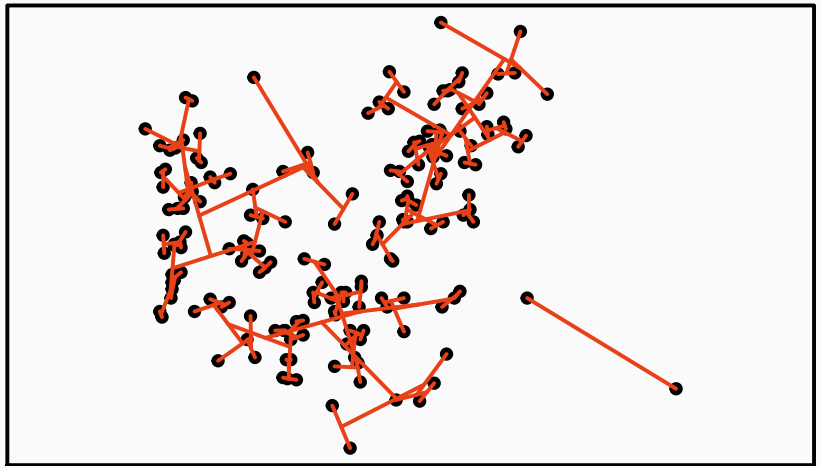
Complete



Single

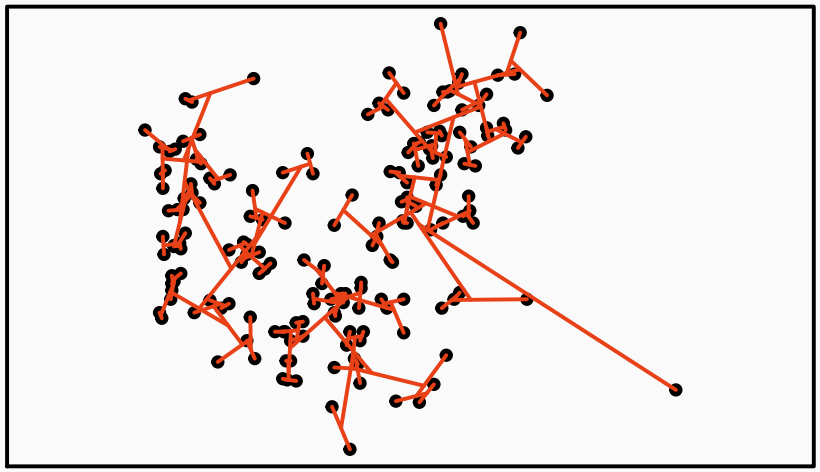


Average

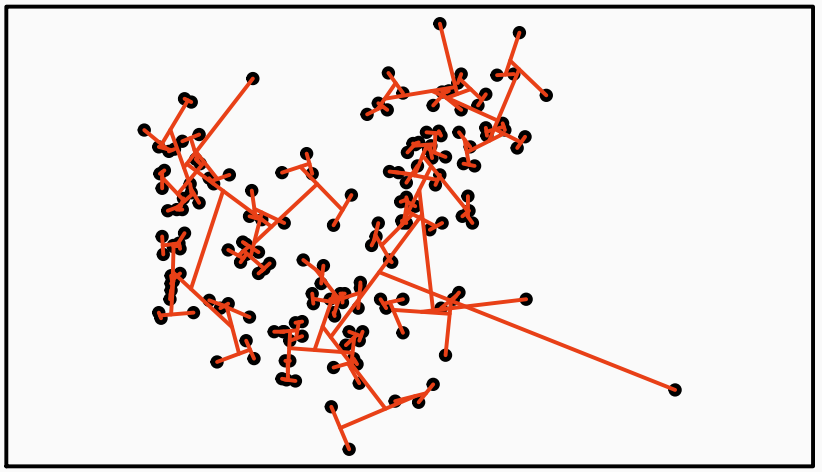


Example

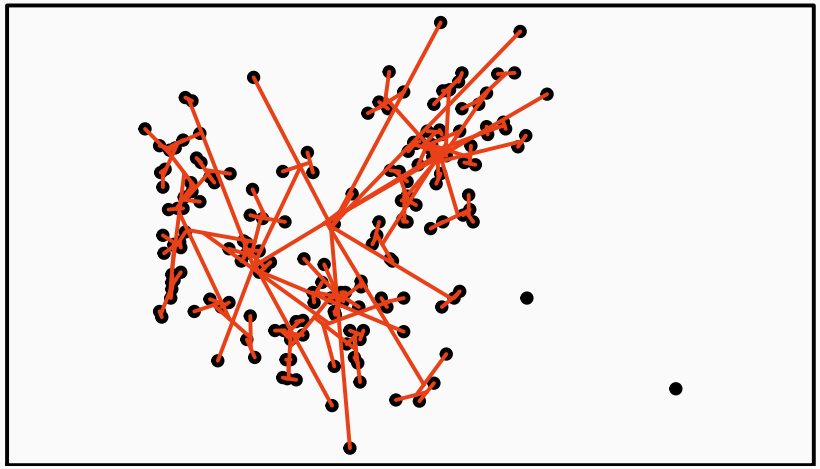
Ward



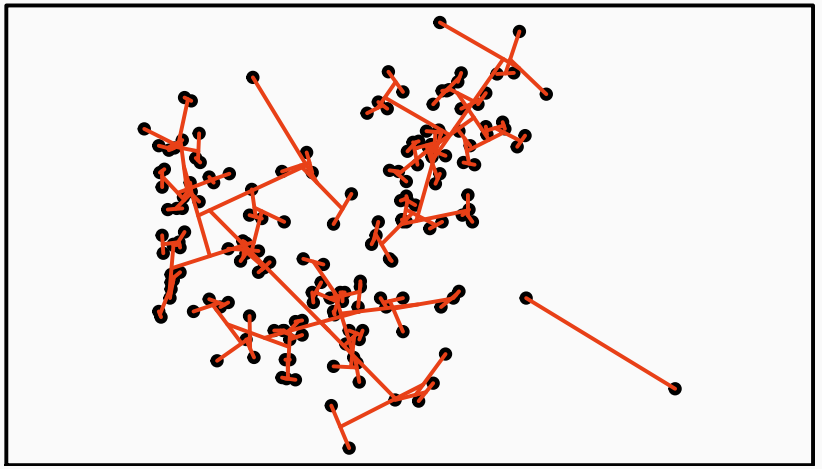
Complete



Single

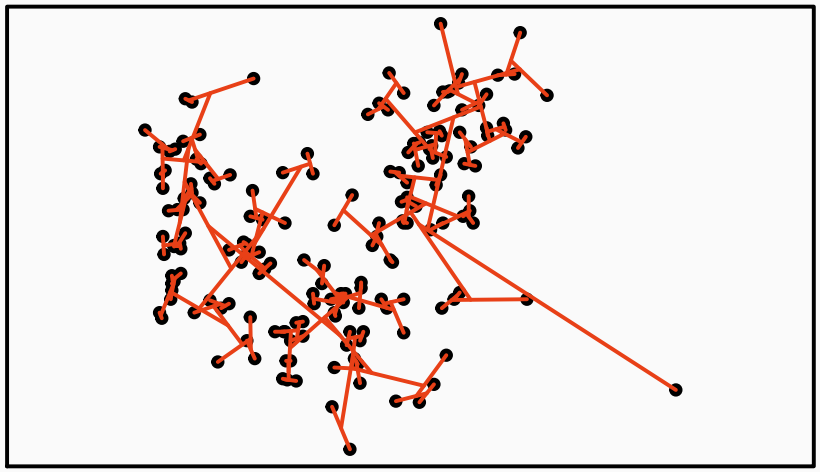


Average

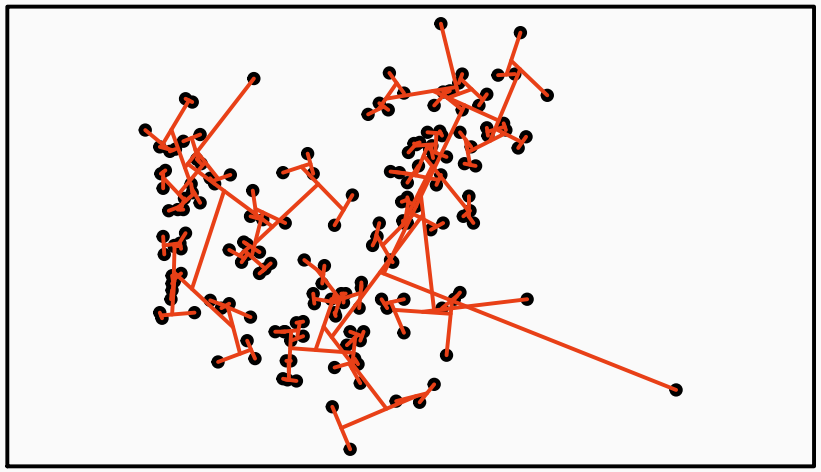


Example

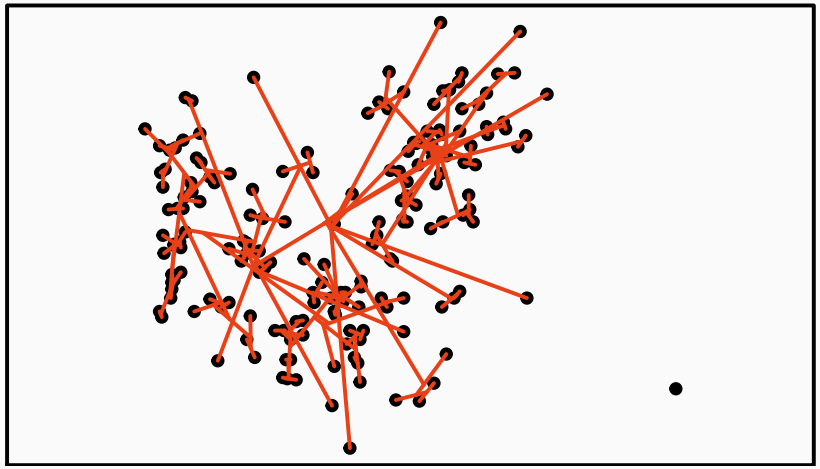
Ward



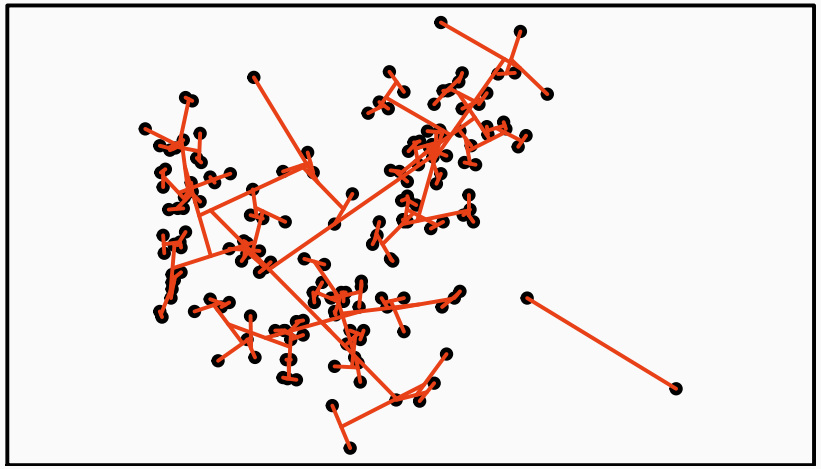
Complete



Single

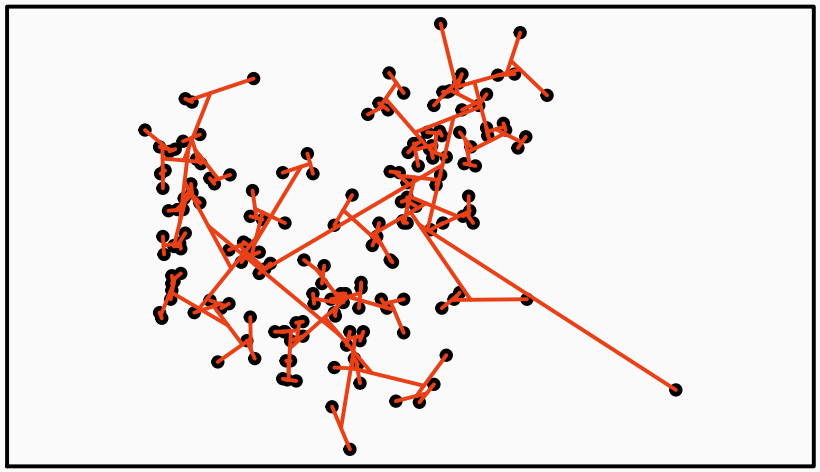


Average

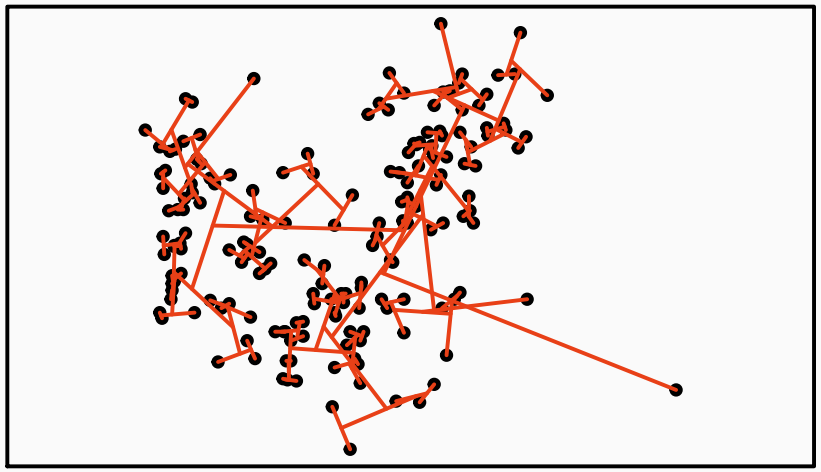


Example

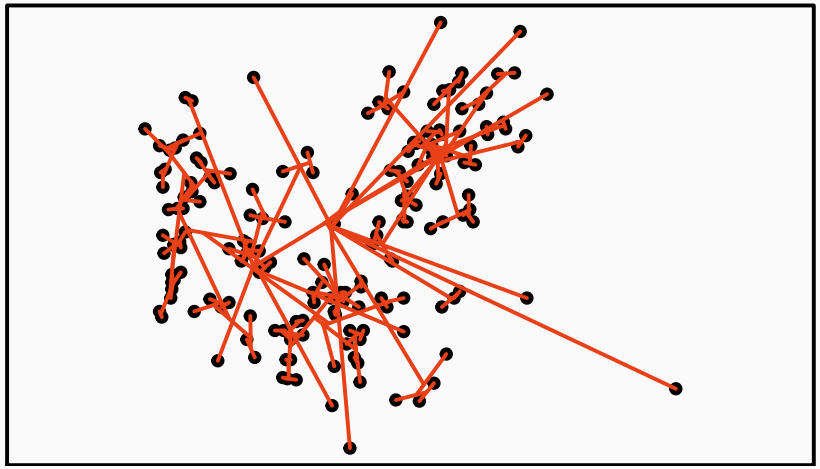
Ward



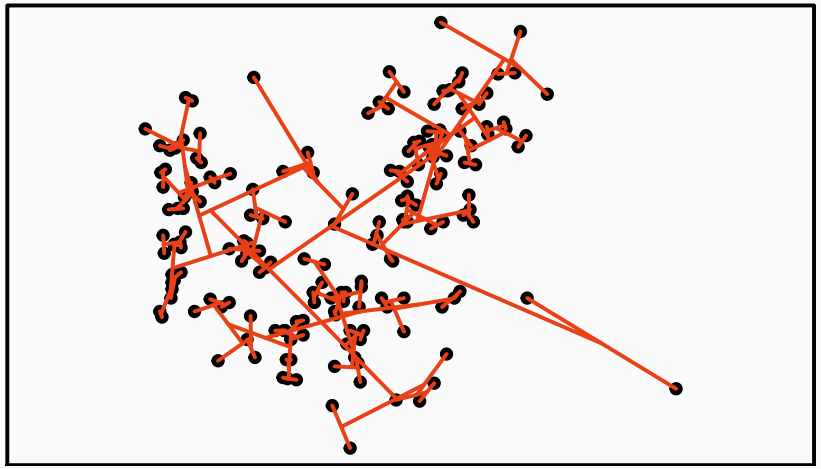
Complete



Single



Average



Discussion

Ward versus average

- somewhat related hierarchy
- Ward's method tend to favor more balanced class size
- Ward's method is closely related to other methods (e.g. K-means)
- outliers are more easily aggregated with other points in the Ward's method

Numerous other variants

- centroid and median linkage (distances between classes induced by prototype based representation)
- minimax linkage (enclosing ball diameter)
- etc.

Divisive methods

Generic algorithm

- initial partition: $\mathcal{P}^1 = \{1, \ldots, N\}$
- for k from 2 to N:
  - chose a class to split
  - build $\mathcal{P}^k$ from $\mathcal{P}^{k-1}$ by splitting the chosen class into two sub classes

Difficulties

- choosing the class to split is relatively easy by using e.g. a diameter criterion
- but splitting is hard: too many possible splits $(2^{N-1} - 1)$!

DIANA

- DIvisive ANAlysis Clustering (available in R in cluster)
- reference algorithm by Kaufman & Roussew, 1990

DIANA

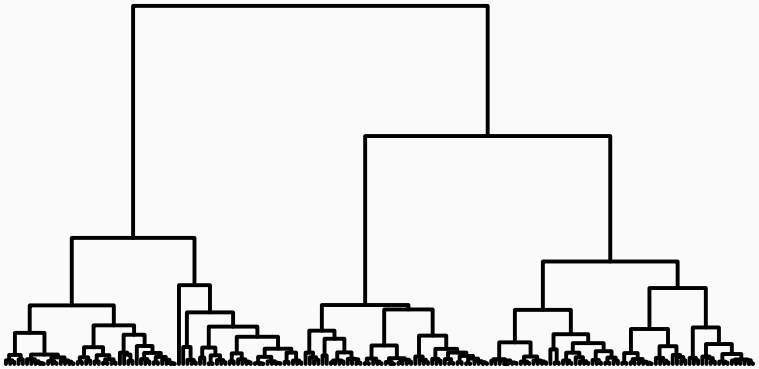
$\mathcal{P}^1 \leftarrow \{1, \ldots, N\}$
for k in $2, \ldots, N$ **do**
  find C_j in $\mathcal{P}^{k-1}$ with the largest diameter
  find $\mathbf{X}_l$ for $l \in C_j$ that maximizes $\sum_{l' \in C_j} d(\mathbf{X}_l, \mathbf{X}_{l'})$
  $C_j^a \leftarrow \{l\}$ and $C_j^b \leftarrow C_j \setminus \{l\}$
  repeat
    for all t in C_j compute

$$D(t) = \frac{1}{|C_j^b|} \sum_{u \in C_j^b} d(\mathbf{X}_t, \mathbf{X}_u) - \frac{1}{|C_j^a|} \sum_{u \in C_j^a} d(\mathbf{X}_t, \mathbf{X}_u)$$

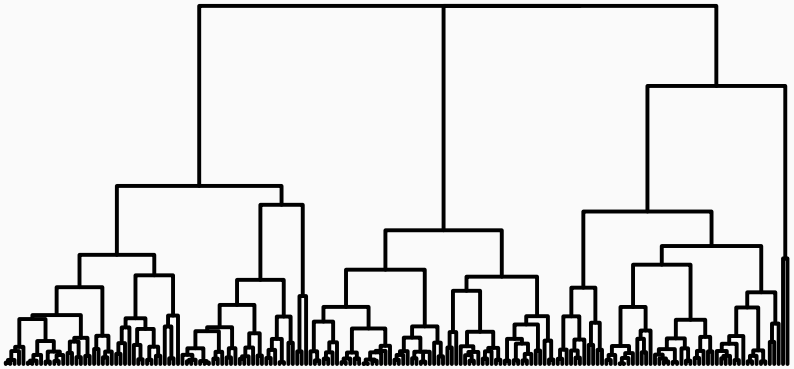
    if $v = \arg\max_t D(t)$ is such that $D(v) > 0$ move v from C_k^b to C_k^a
  until $\max_t D(t) < 0$
  define $\mathcal{P}^k$ as $\mathcal{P}^{k-1}$ in which C_j is replaced by C_j^a and C_j^b
end for

Example

Ward

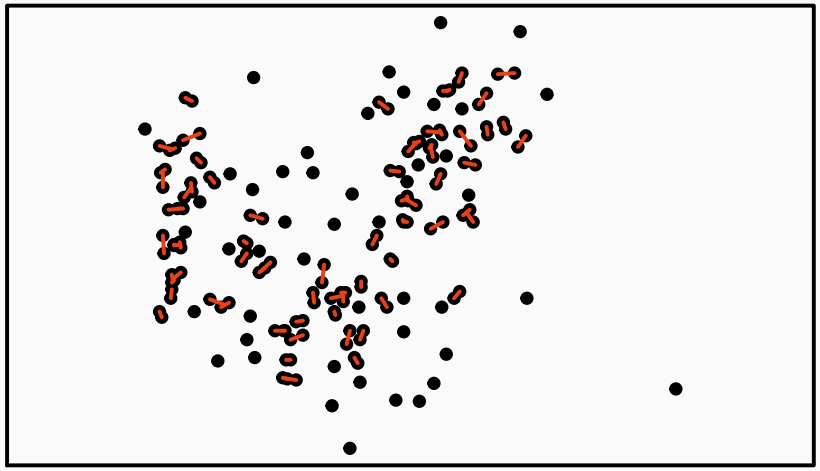


Diana

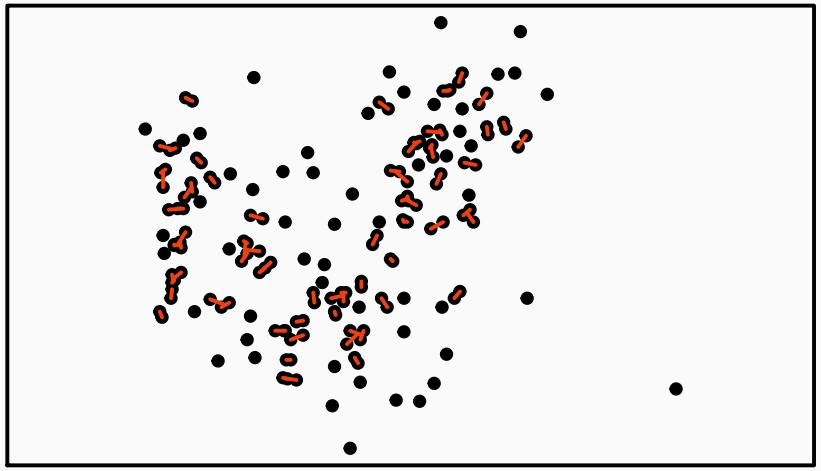


Example

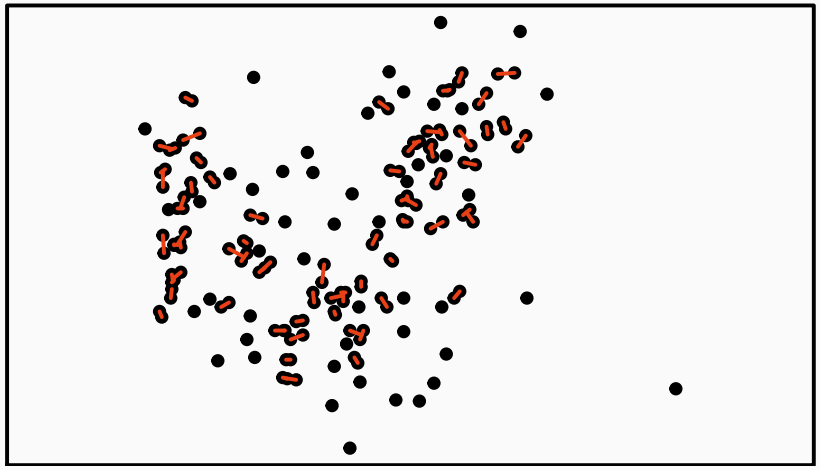
Ward



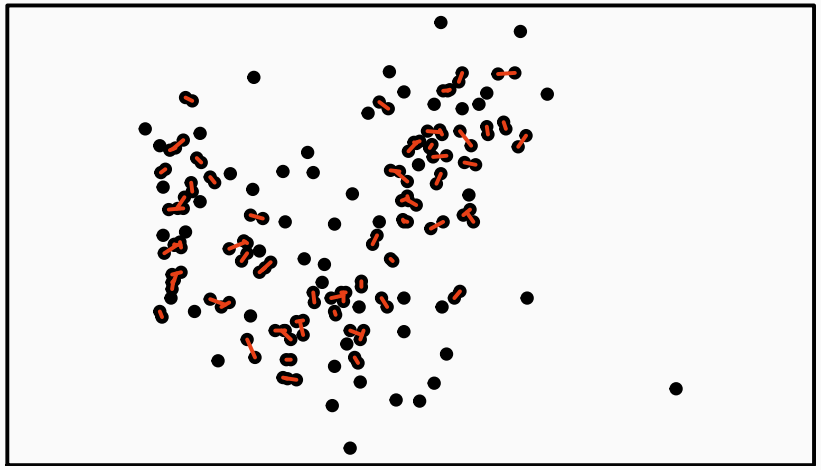
Average



Complete

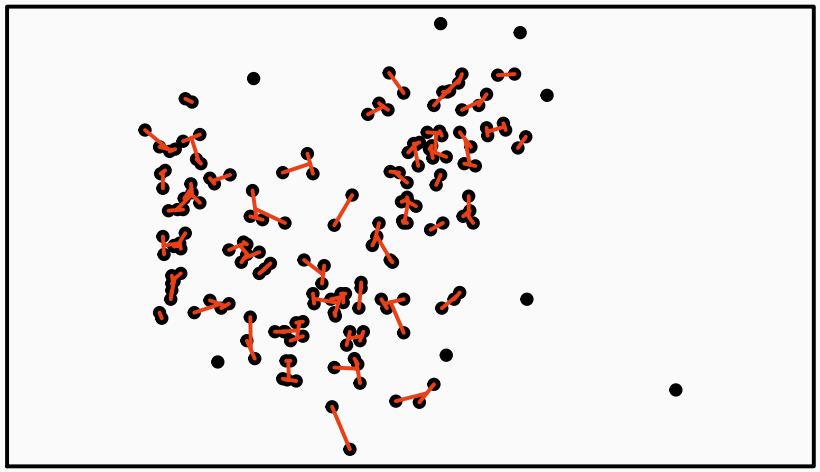


Diana

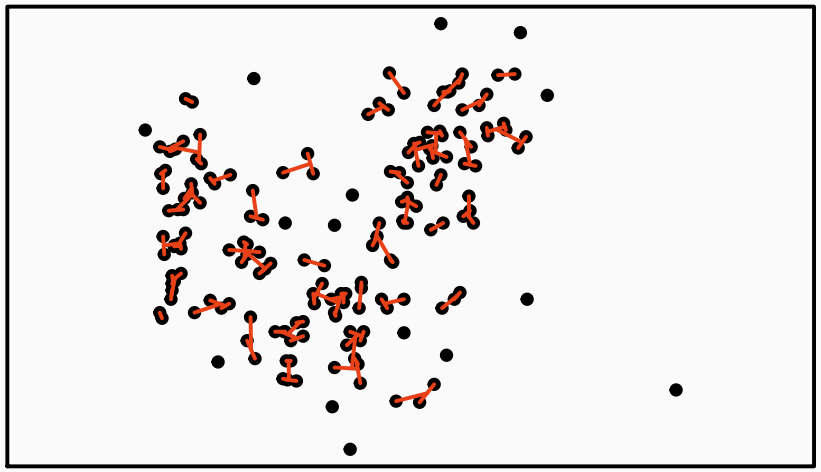


Example

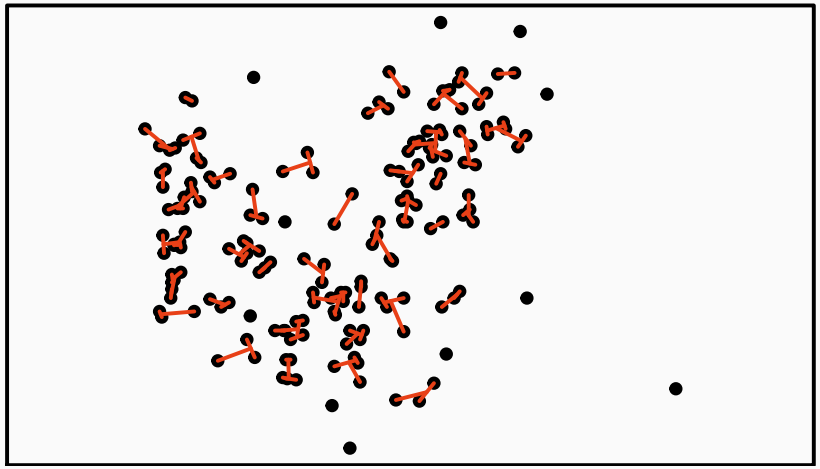
Ward



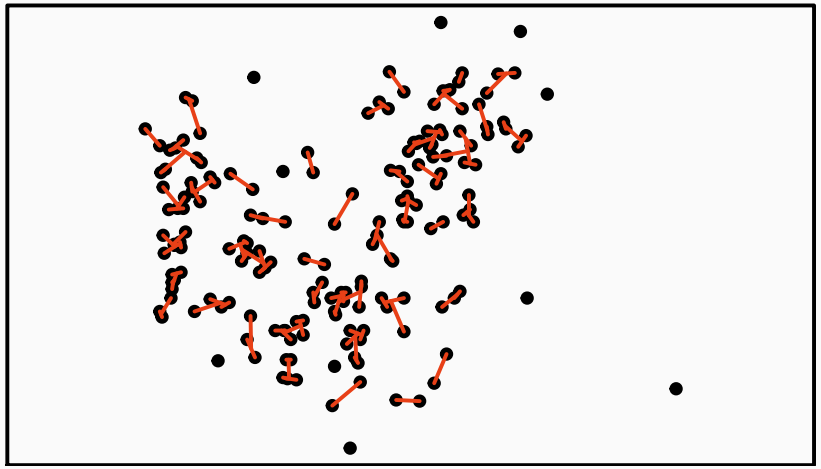
Average



Complete

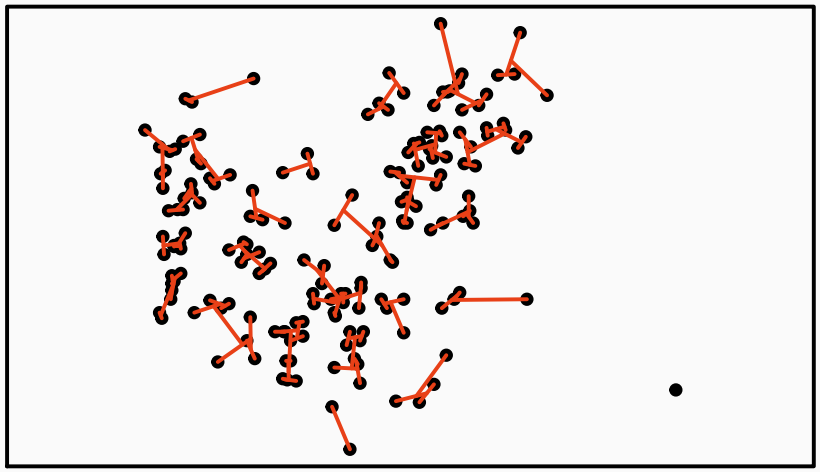


Diana

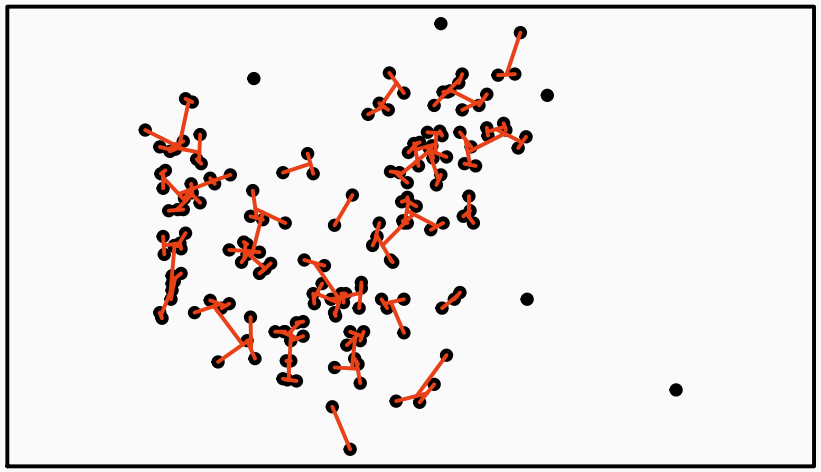


Example

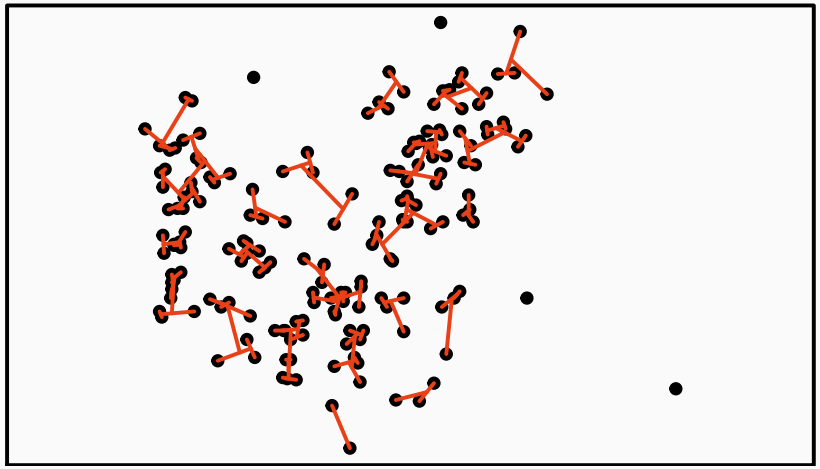
Ward



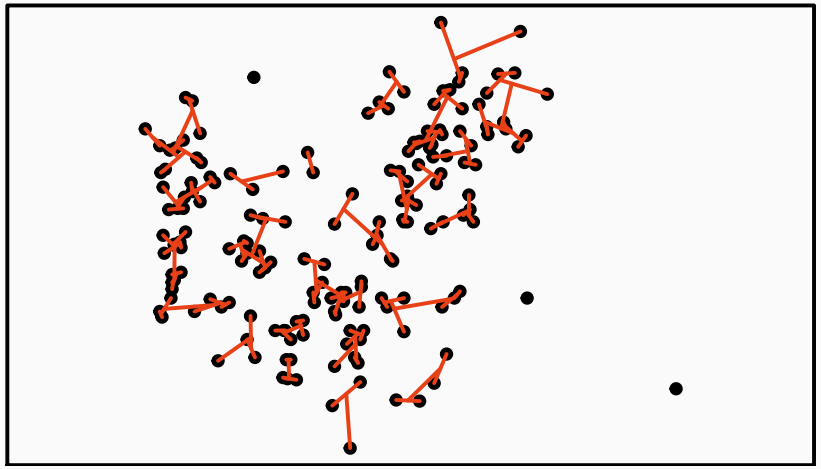
Average



Complete

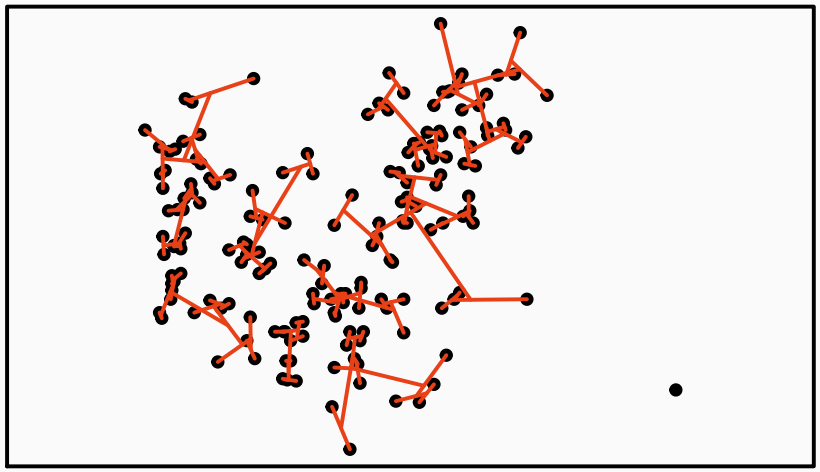


Diana

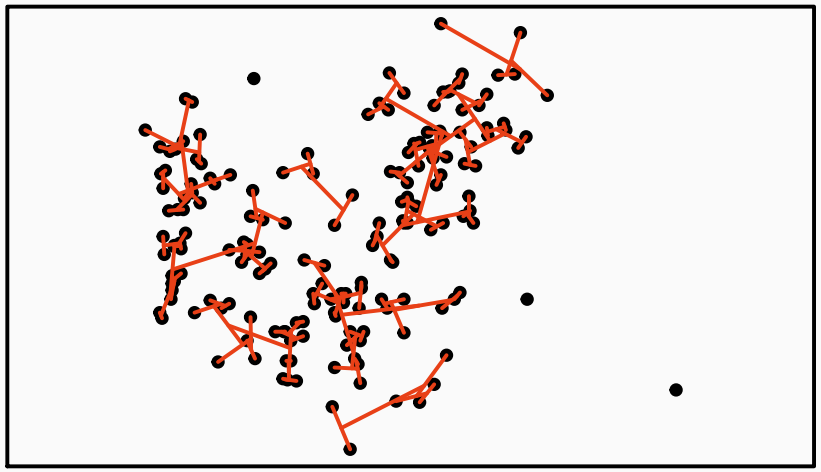


Example

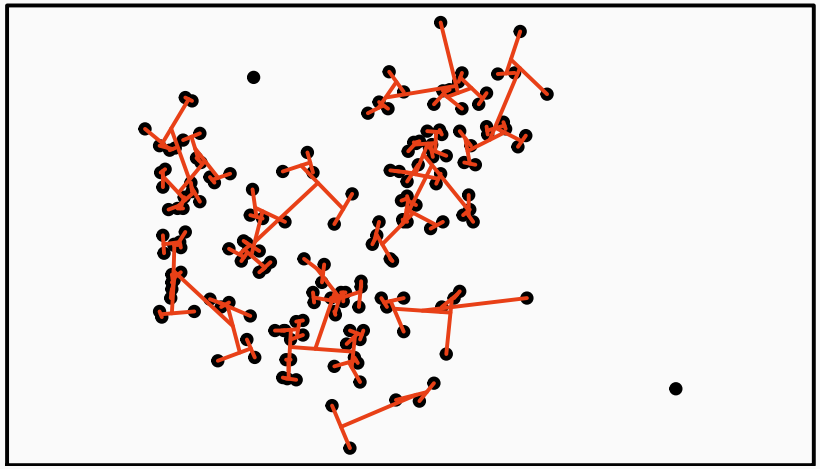
Ward



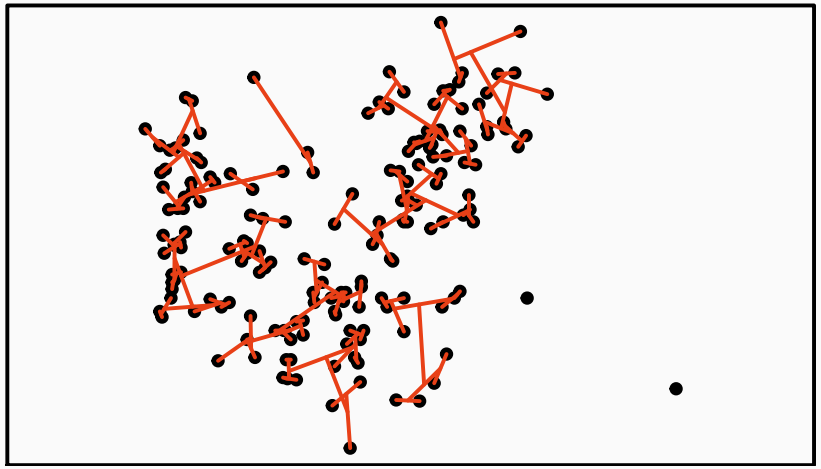
Average



Complete

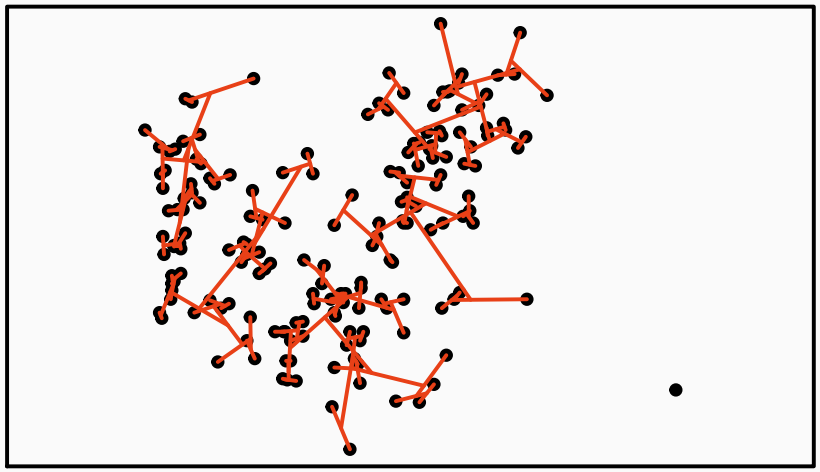


Diana

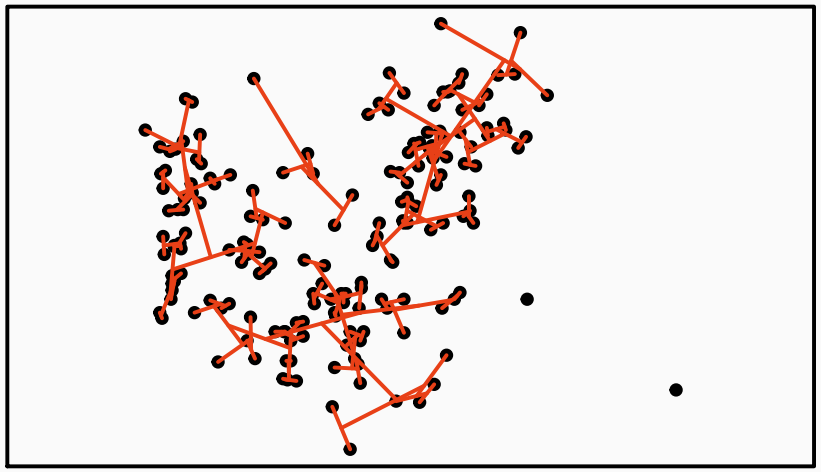


Example

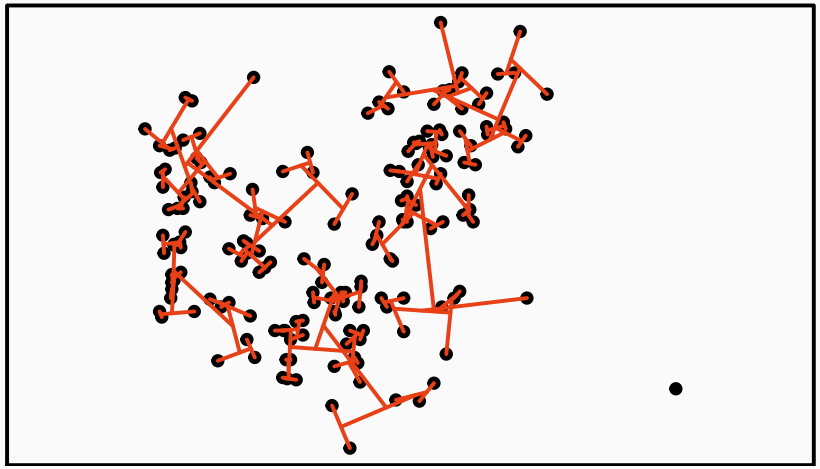
Ward



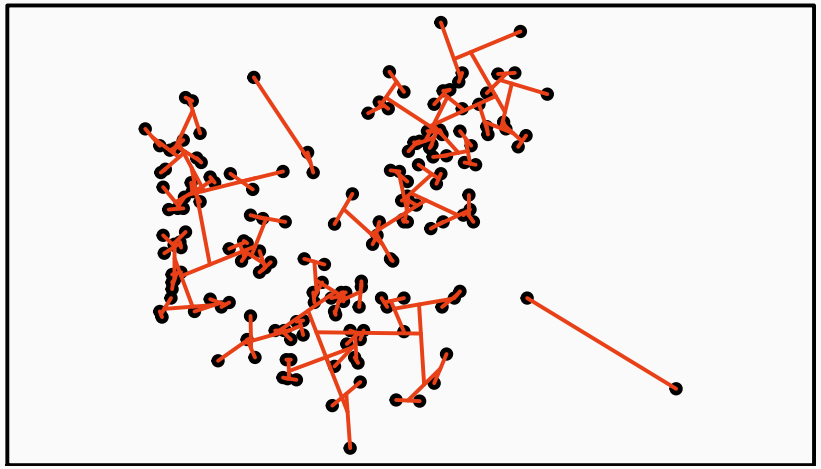
Average



Complete

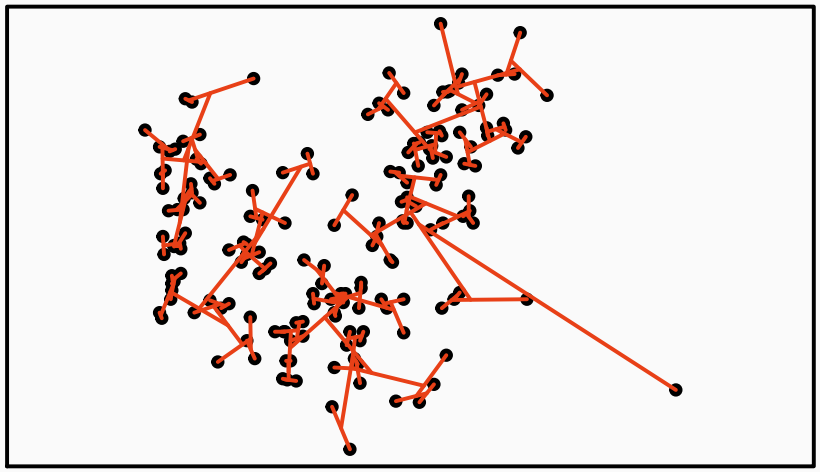


Diana

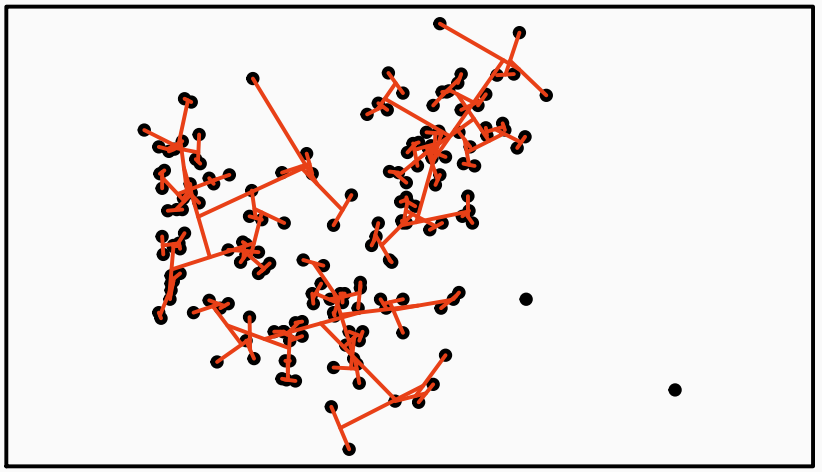


Example

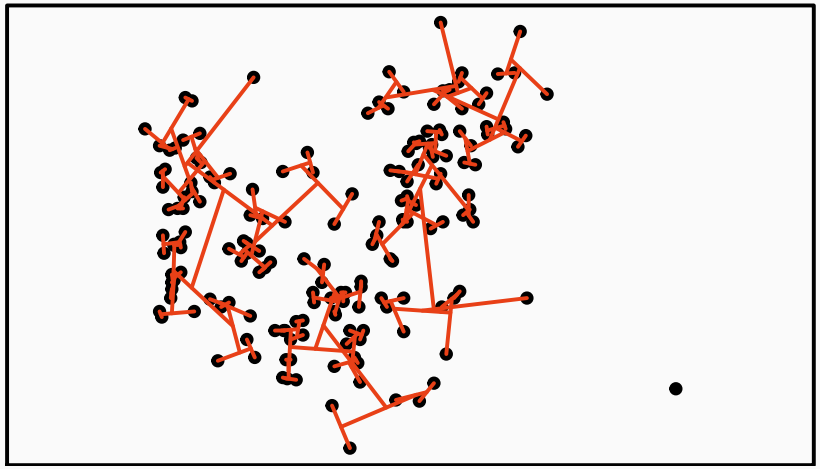
Ward



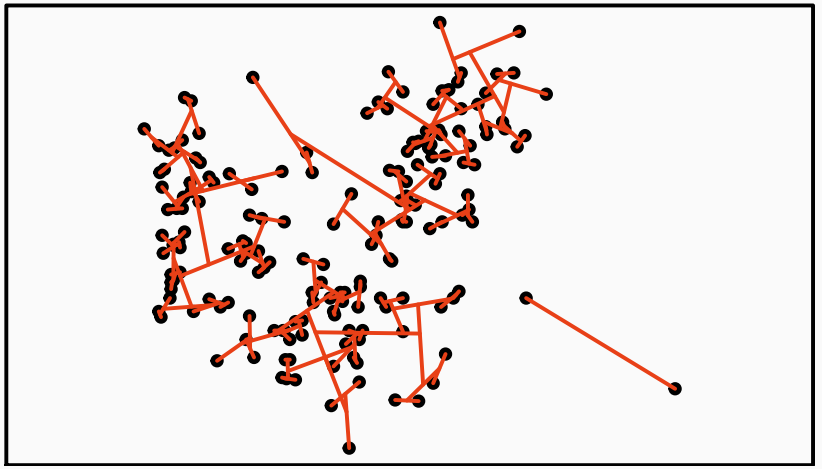
Average



Complete

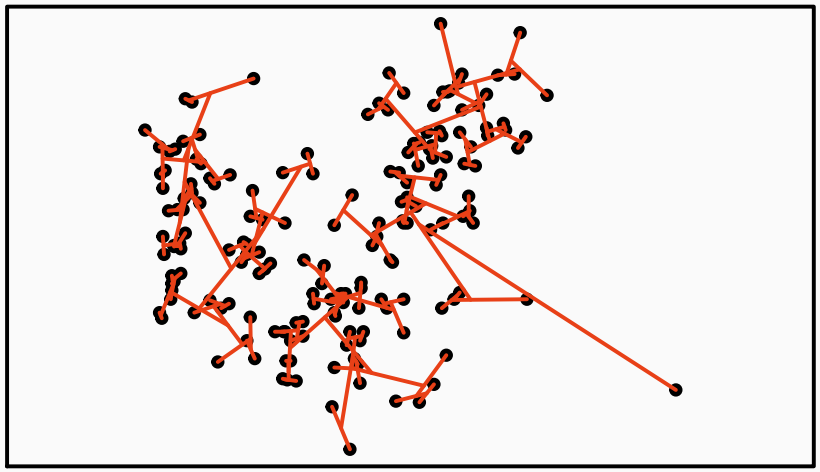


Diana

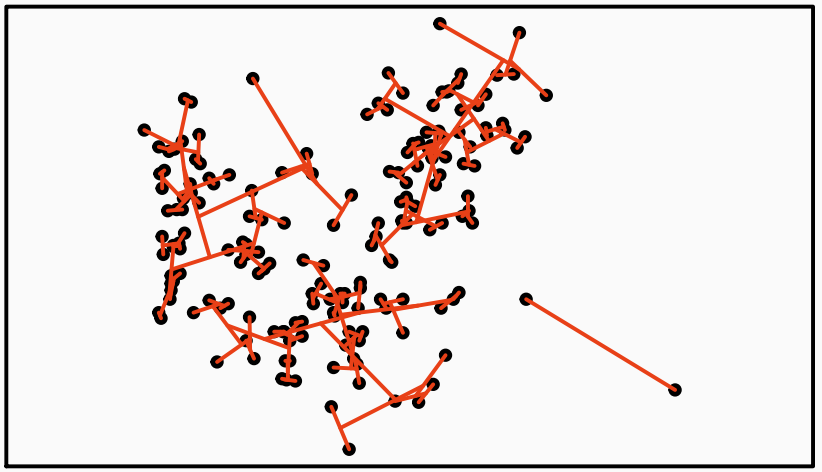


Example

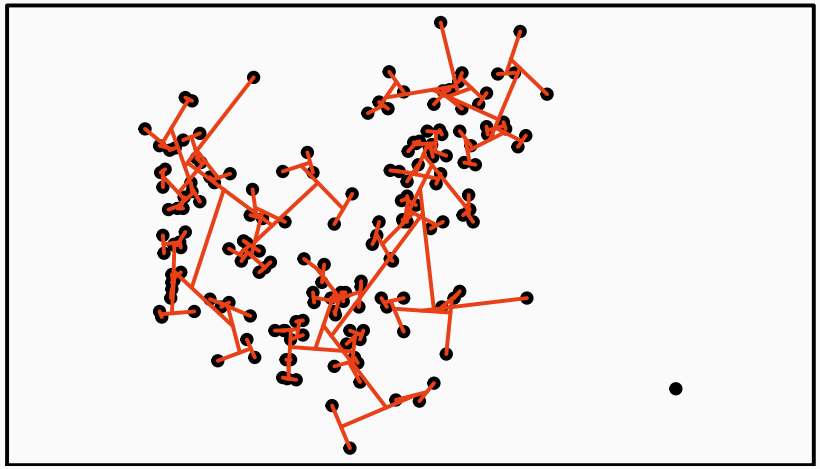
Ward



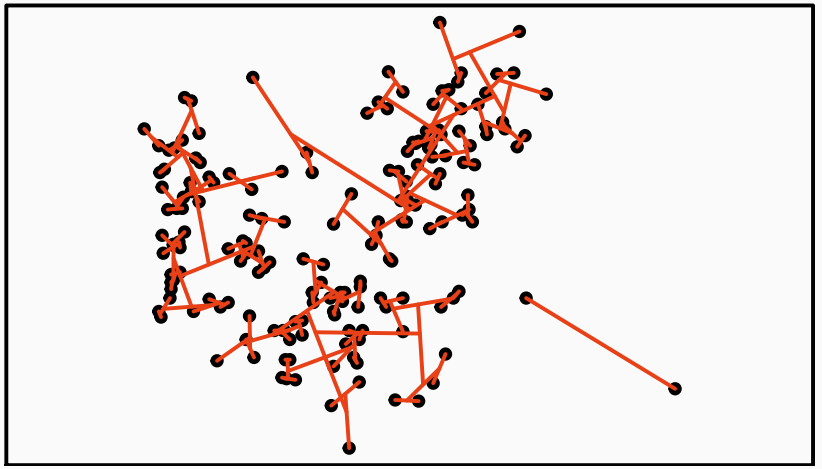
Average



Complete

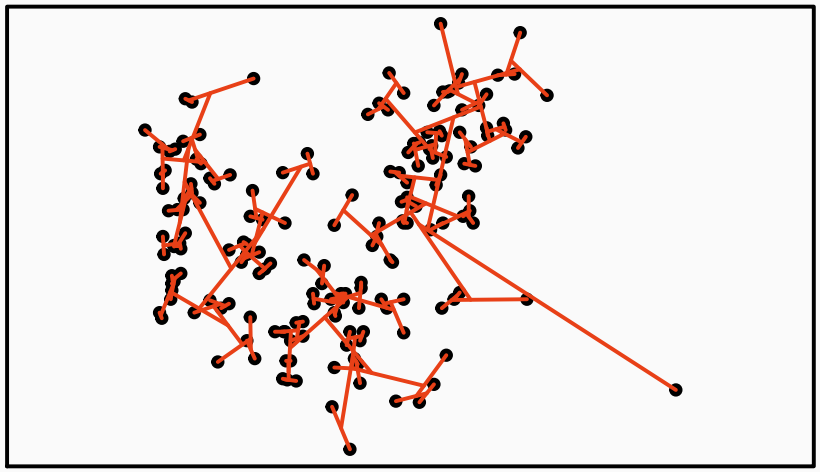


Diana

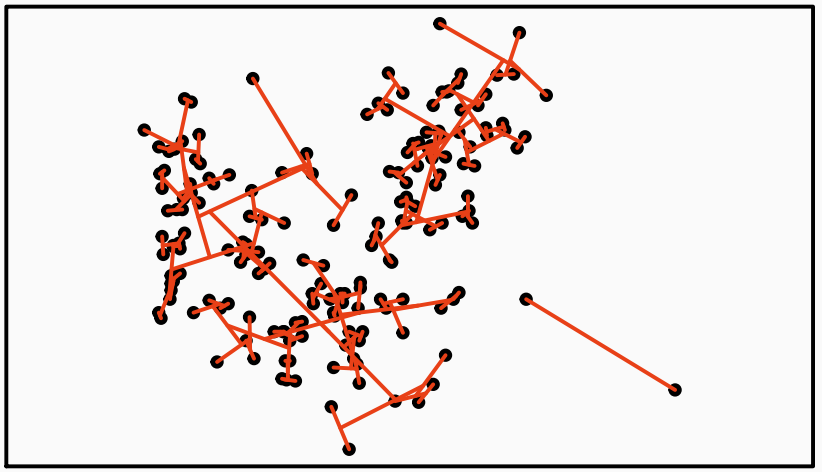


Example

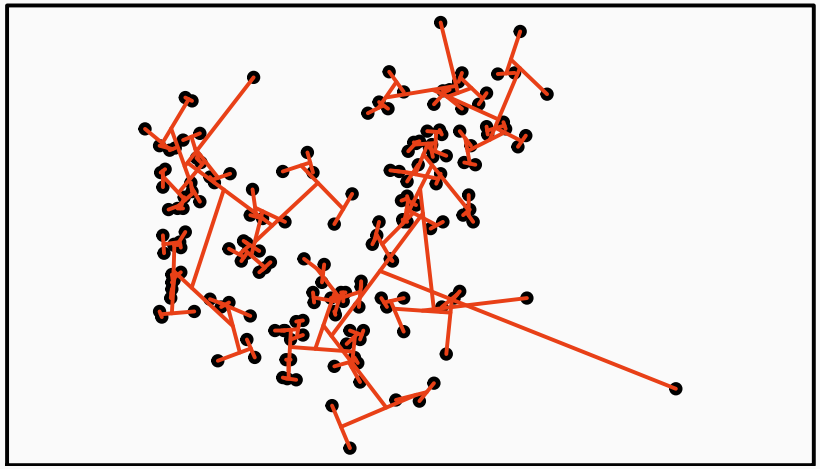
Ward



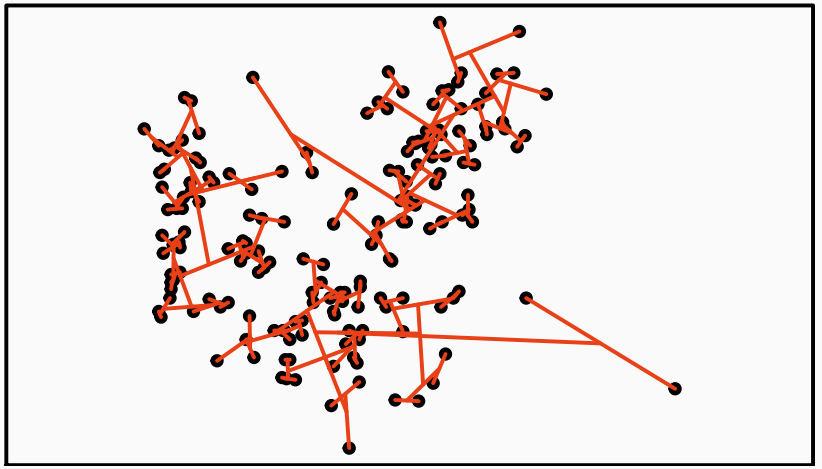
Average



Complete

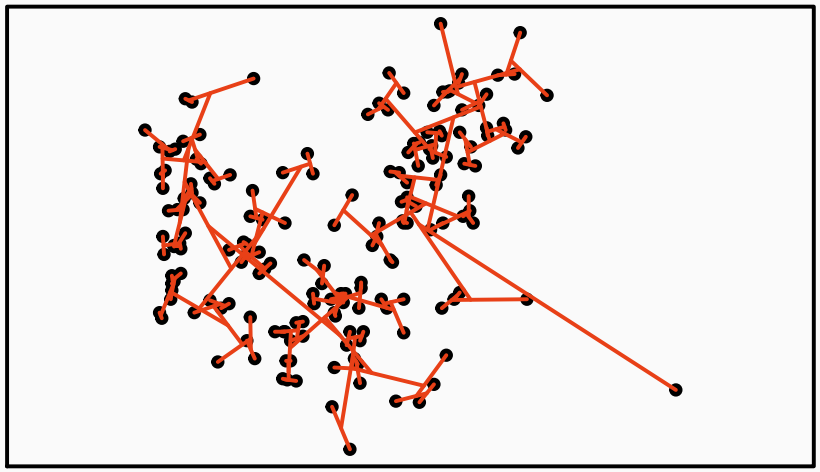


Diana

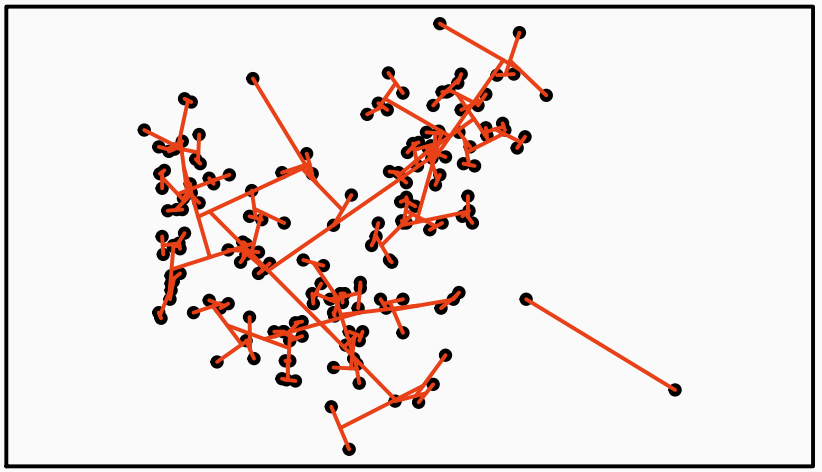


Example

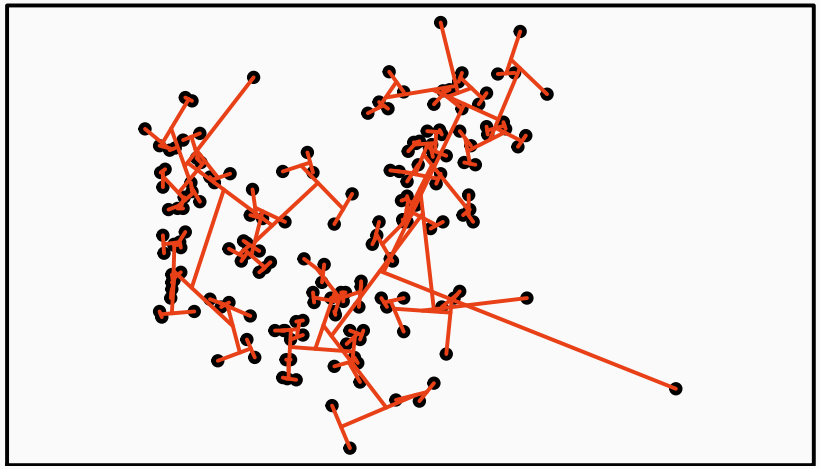
Ward



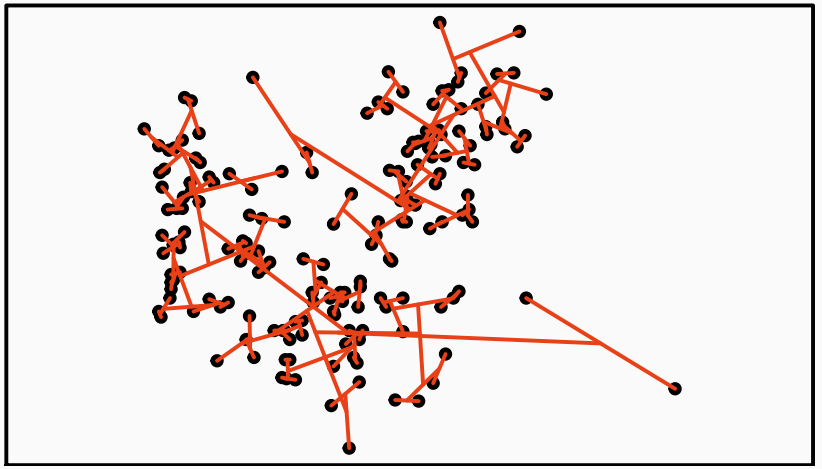
Average



Complete

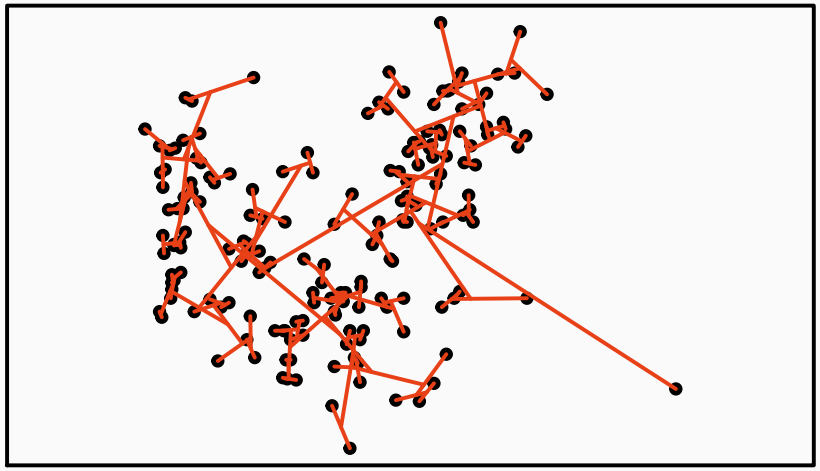


Diana

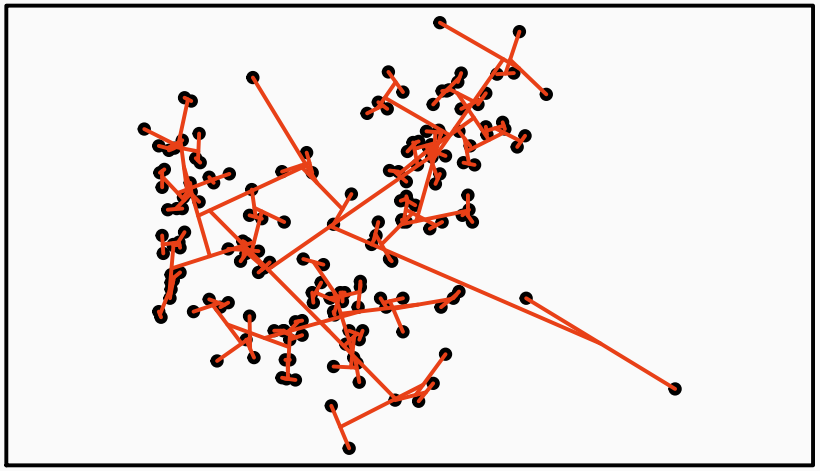


Example

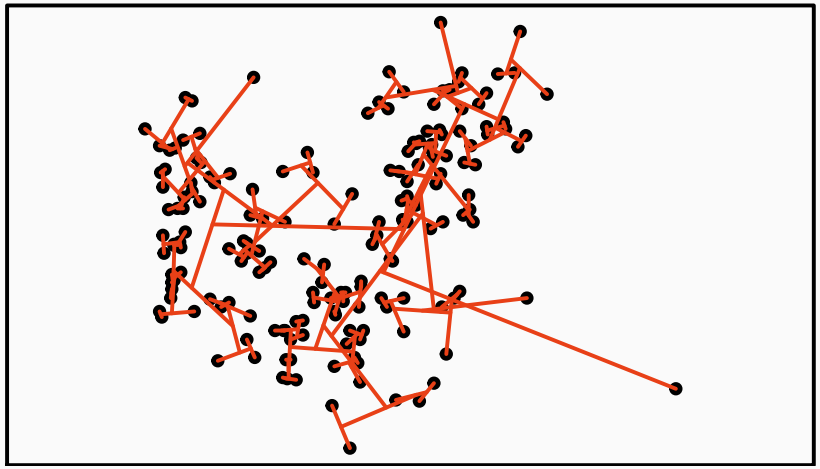
Ward



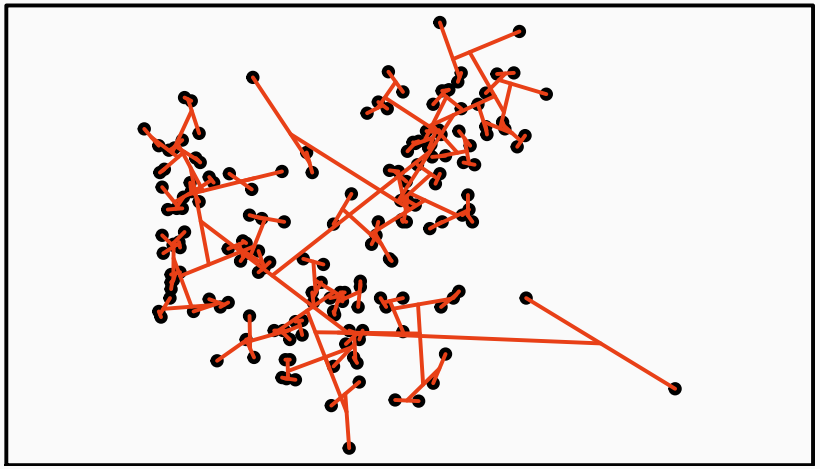
Average



Complete



Diana



Summary

Hierarchical clustering

- ☺ provide a hierarchy and a dendrogram
- ☺ applicable to any dissimilarity
- ☹ $\Theta\left(N^2\right)$ running time for aggregative methods
- ☹ not much guarantees about the final result
- ☹ inferior results for some methods (e.g. single linkage)

Generic questions

- dissimilarity?
- hierarchical method?
- cluster number?

Outline

Criterion based clustering

Optimization point of view

- define a quality criterion for a clustering structure
- optimize this quality over “all” clustering structures
- typical example:
  - within variance as quality measure
  - optimized over partitions

Difficulties

- no obvious quality criterion for exploratory tasks
- very difficult discrete optimization problems (NP-hard in some cases)

Quantization point of view

Summarizing a data set

- key idea: represent a data set $\mathcal{D} = (\mathbf{X}_i)_{1 \leq i \leq N}$ by a smaller set of **prototypes** $\mathcal{D} = (\gamma_k)_{1 \leq k \leq K}$
- $\mathbf{X}_i$ is presented by a prototype: z_i is the index in $\{1, \ldots, K\}$ of this prototype
- natural risk associated to the problem

$$\mathcal{E}(\mathbf{\Gamma}, \mathbf{z}) = \sum_{i=1}^{N} d(\mathbf{X}_i, \gamma_{z_i}),$$

where $\mathbf{z} = (z_1, \ldots, z_N)$ and $\mathbf{\Gamma} = (\gamma_1, \ldots, \gamma_K)$.

Quantization point of view

Summarizing a data set

- key idea: represent a data set $\mathcal{D} = (\mathbf{X}_i)_{1 \leq i \leq N}$ by a smaller set of **prototypes** $\mathcal{D} = (\gamma_k)_{1 \leq k \leq K}$
- $\mathbf{X}_i$ is presented by a prototype: z_i is the index in $\{1, \ldots, K\}$ of this prototype
- natural risk associated to the problem

$$\mathcal{E}(\mathbf{\Gamma}, \mathbf{z}) = \sum_{i=1}^{N} d(\mathbf{X}_i, \gamma_{z_i}),$$

  where $\mathbf{z} = (z_1, \ldots, z_N)$ and $\mathbf{\Gamma} = (\gamma_1, \ldots, \gamma_K)$.
- natural clustering interpretation: $\mathbf{z}$ defines a partition of $\{1, \ldots, N\}$ into K classes!

Optimization

Difficult!

- minimizing $\mathcal{E}(\mathbf{\Gamma}, \mathbf{z})$ is a combinatorial problem
- for standard d (such as the squared Euclidean distance when $\mathcal{X} = \mathbb{R}^P$) this is NP-hard

Simple sub-problems

- minimizing $\mathcal{E}(\mathbf{\Gamma}, \mathbf{z})$ with respect to $\mathbf{z}$ is easy

$$\arg\min_{\mathbf{z}} \mathcal{E}(\mathbf{\Gamma}, \mathbf{z}) = \left(\arg\min_{k \in \{1,\dots,K\}} d(\mathbf{X}_1, \gamma_k), \dots, \arg\min_{k \in \{1,\dots,K\}} d(\mathbf{X}_N, \gamma_k) \right)$$

- if prototypes are restricted to be elements of the data set $\mathcal{D}$, optimizing with respect to $\mathbf{\Gamma}$ is also easy

$$\arg\min_{\mathbf{\Gamma}} \mathcal{E}(\mathbf{\Gamma}, \mathbf{z}) = \left(\arg\min_{\gamma_k \in \mathcal{D}} \sum_{i, z_i = k} d(\mathbf{X}_i, \gamma_k) \right)_{1 \leq k \leq K}$$

Coordinate descent

A.k.a. alternating optimization

a simple minimization algorithm for a function $F(u, v)$:

select u_0 randomly
$k \leftarrow 1$
repeat
    $v_k = \arg\min_v F(u_{k-1}, v)$
    $u_k = \arg\min_u F(u, v_k)$
    $k \leftarrow k + 1$
until convergence

Properties

- converges to a local minimum
- but not to a global one
- improved by multiple restarts

K-medoids

Algorithm

select $\boldsymbol{\Gamma}$ as a random subset of $\mathcal{D}$
repeat
  $z_i \leftarrow \arg\min_{k \in \{1,\ldots,K\}} d(\mathbf{X}_i, \gamma_k)$ ▷ assignment phase
  $\gamma_k \leftarrow \arg\min_{\gamma \in \mathcal{D}} \sum_{i, z_i = k} d(\mathbf{X}_i, \gamma)$ ▷ representation phase
until convergence

Comments

- one of the most well known clustering algorithm for arbitrary dissimilarities
- complexity $\Theta(NK) + \Theta\left(N^2\right)$ (assuming d is known)
- complex quantization effects when N is small ("data holes")

Euclidean/Hilbert case

When $\mathcal{X} = \mathbb{R}^P$

- one uses in general $d(\mathbf{X}_i, \mathbf{X}_j) = \|\mathbf{X}_i - \mathbf{X}_j\|^2$
- then we do need to restrict γ to elements of $\mathcal{D}$
  - we have
$$\mathcal{E}(\mathbf{\Gamma}, \mathbf{z}) = \sum_{k=1}^{K} \sum_{i, z_i = k} \|\mathbf{X}_i - \gamma_k\|^2$$
  - and thus
$$\arg\min_{\mathbf{\Gamma}} \mathcal{E}(\mathbf{\Gamma}, \mathbf{z}) = \left(\frac{1}{s_k} \sum_{i, z_i = k} \mathbf{X}_i \right)_{1 \leq k \leq K},$$
    where $s_k = |\{i | z_i = k\}|$
- identical results when $\mathcal{X}$ is a Hilbert space (a RHKS for instance)

K-means algorithm

Algorithm

select $\mathbf{\Gamma}$ as a random subset of $\mathcal{D}$
repeat
  $z_i \leftarrow \arg\min_{k \in \{1,\dots,K\}} \|\mathbf{X}_i - \gamma_k\|^2$ ▷ assignment phase
  $\gamma_k \leftarrow \frac{1}{s_k} \sum_{i, z_i = k} \mathbf{X}_i$ ▷ representation phase
until convergence

Comments

- one of the most well known clustering algorithm
- complexity $\Theta(NKP)$
- numerous variants and improvements (kmeans++ for instance)

Initialization

Standard solution

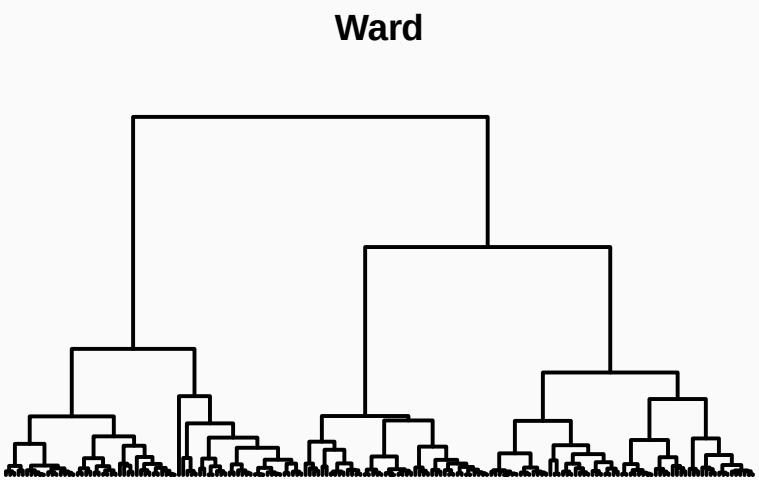
- random prototypes chosen uniformly at random in the data set without replacement
- random clusters do not work properly

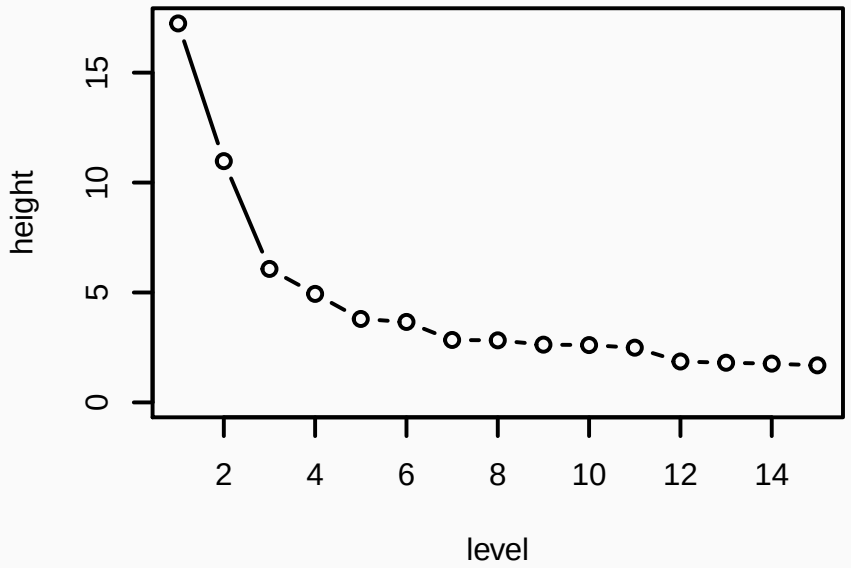
k-means++ (Arthur & Vassilvitskii, 2007 [1])

1. γ_1 chosen uniformly at random in the data set
2. for k in 2 to K:
   - 2.1 for $\mathbf{X}_i \in \mathcal{D} \setminus \{\gamma_1, \ldots, \gamma_{k-1}\}$ compute $p_i = \frac{\min_{k \in \{1,\ldots,k-1\}} d(\mathbf{X}_i, \gamma_k)^2}{\sum_{j \neq i} \min_{k \in \{1,\ldots,k-1\}} d(\mathbf{X}_j, \gamma_k)^2}$
   - 2.2 chose γ_k in $\mathcal{D} \setminus \{\gamma_1, \ldots, \gamma_{k-1}\}$ according to the probabilities $(p_i)_i$

- theoretical guarantees
- practical efficiency

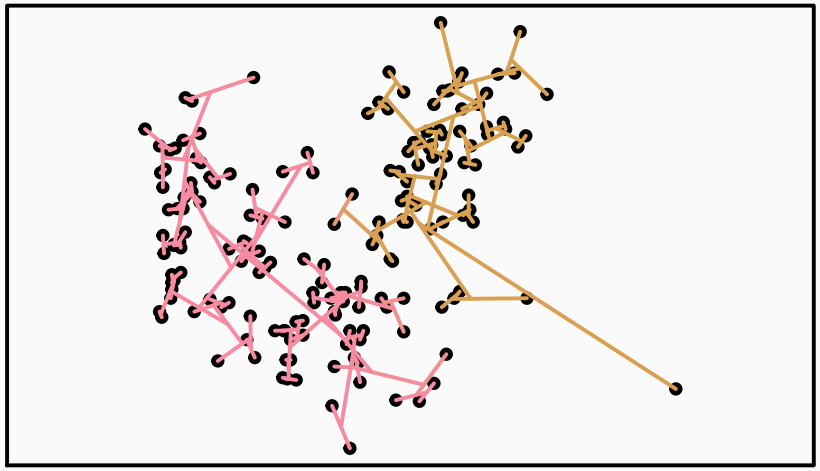
Example



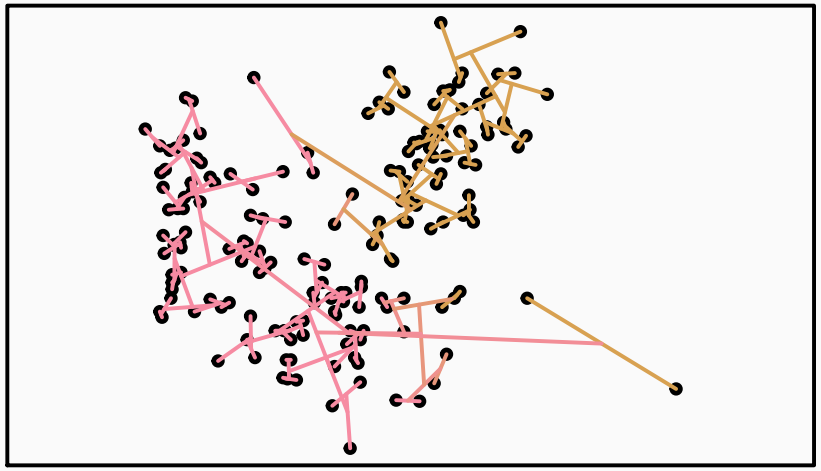


Example

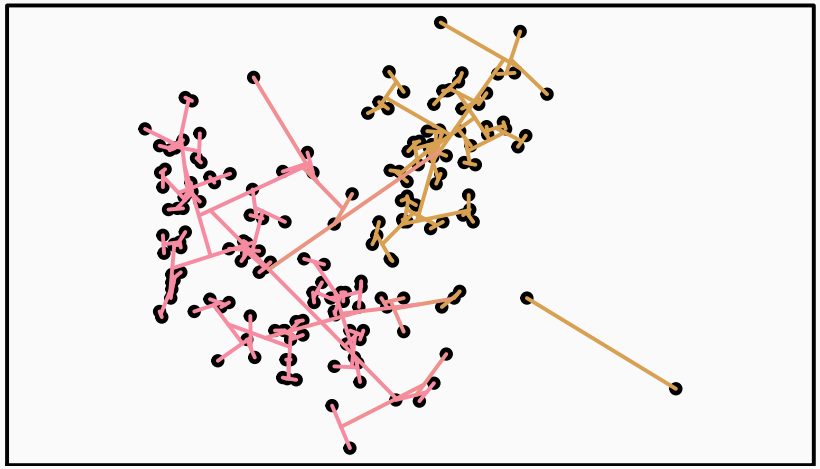
Ward



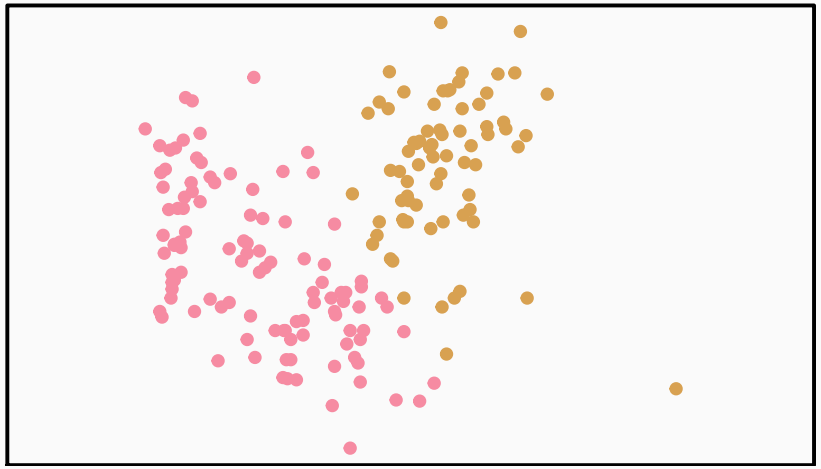
Diana



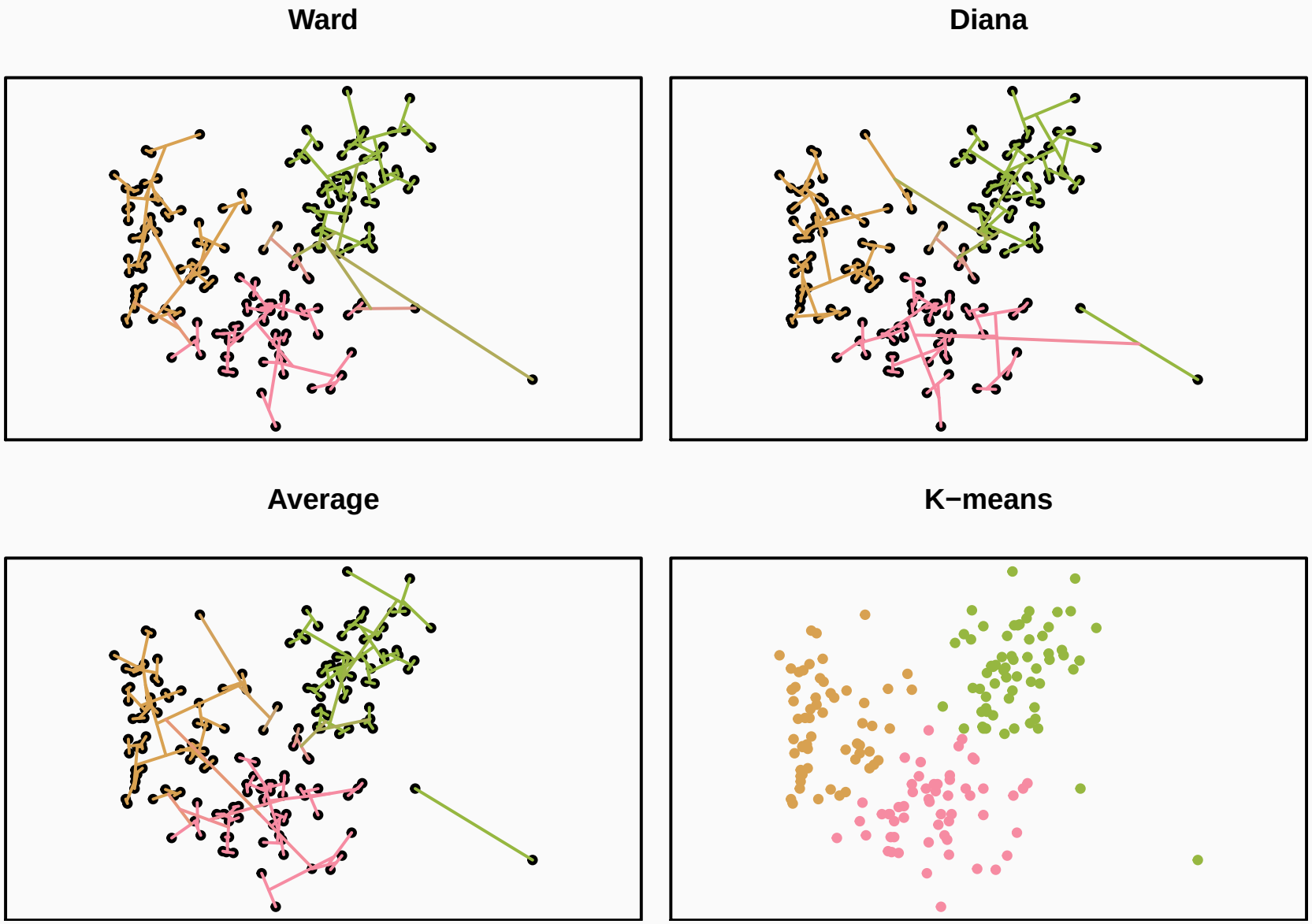
Average



K−means

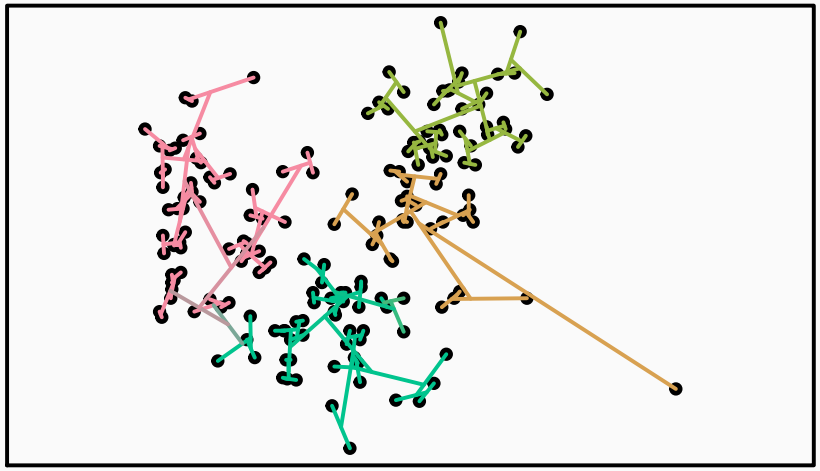


Example

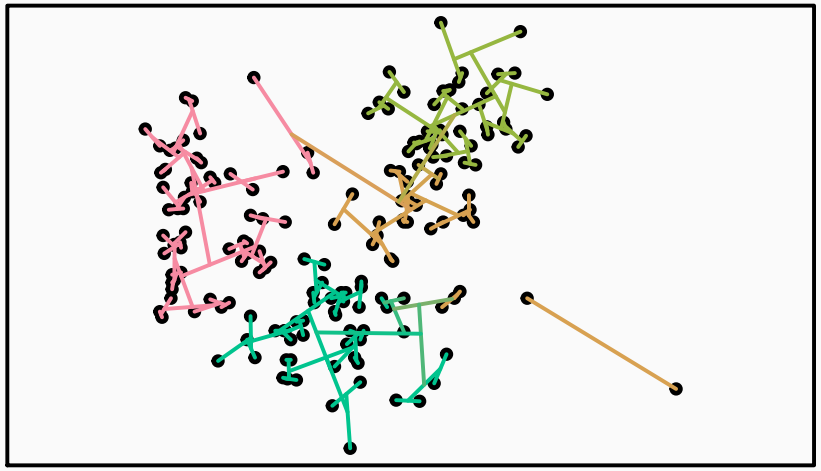


Example

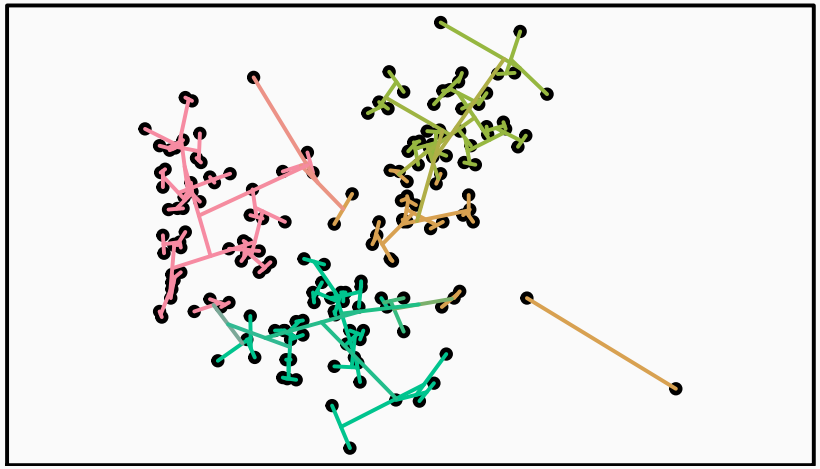
Ward



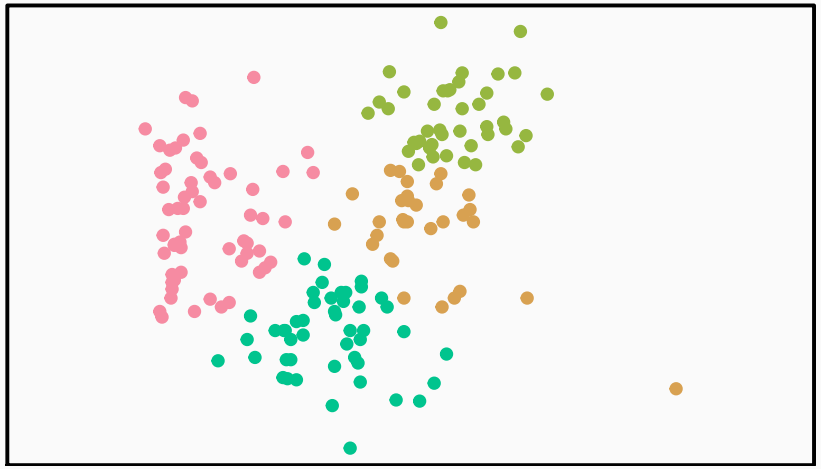
Diana



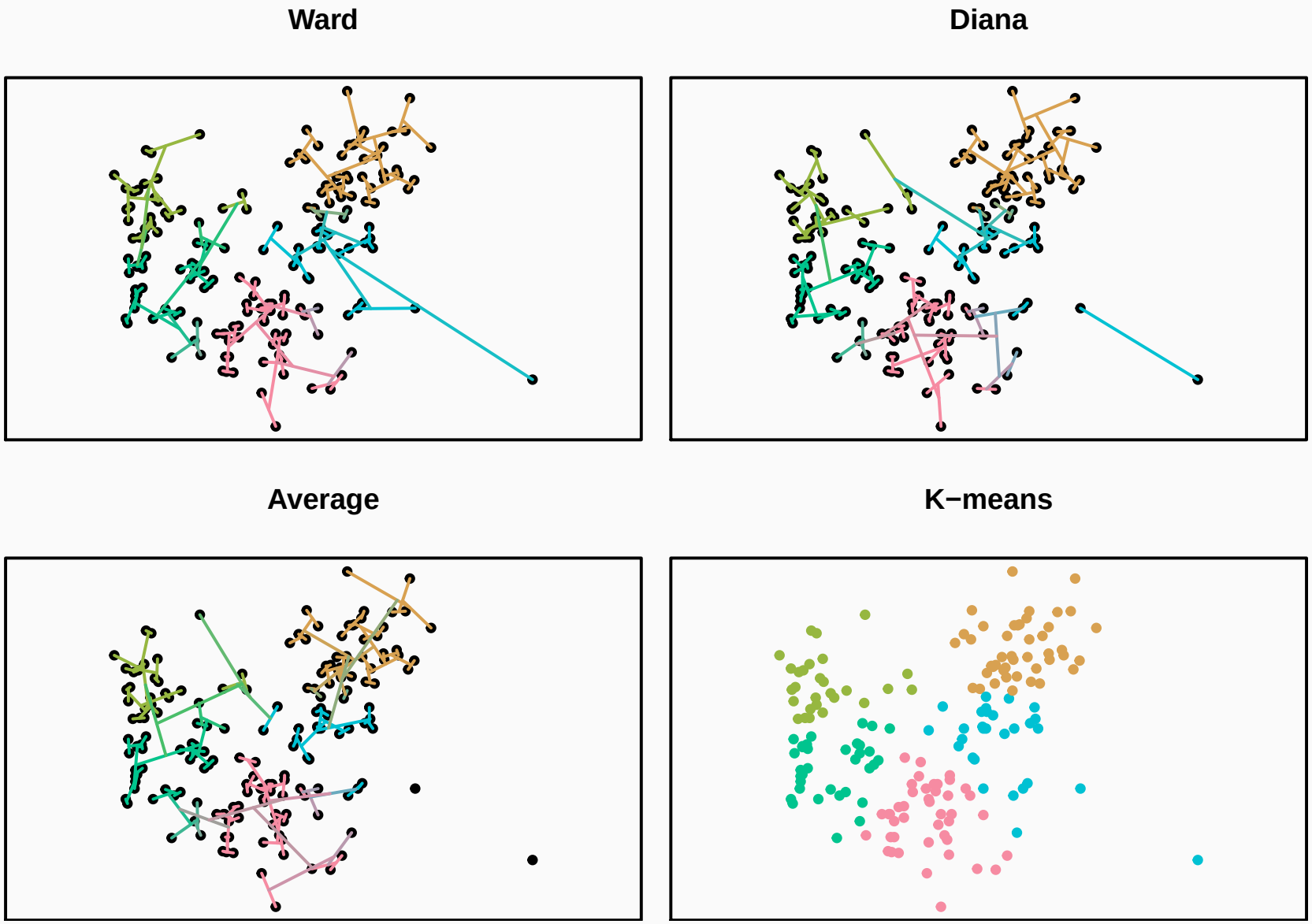
Average



K−means

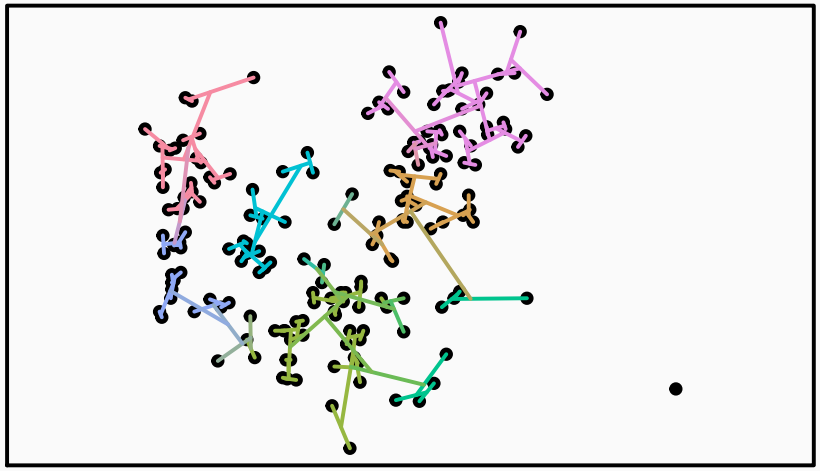


Example

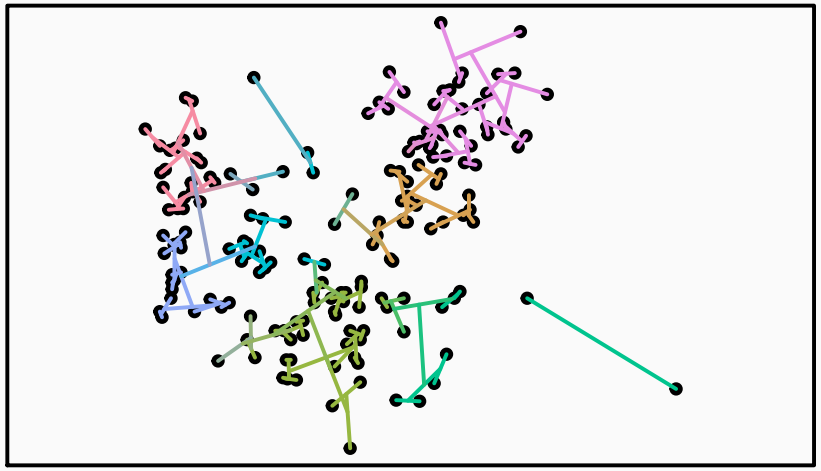


Example

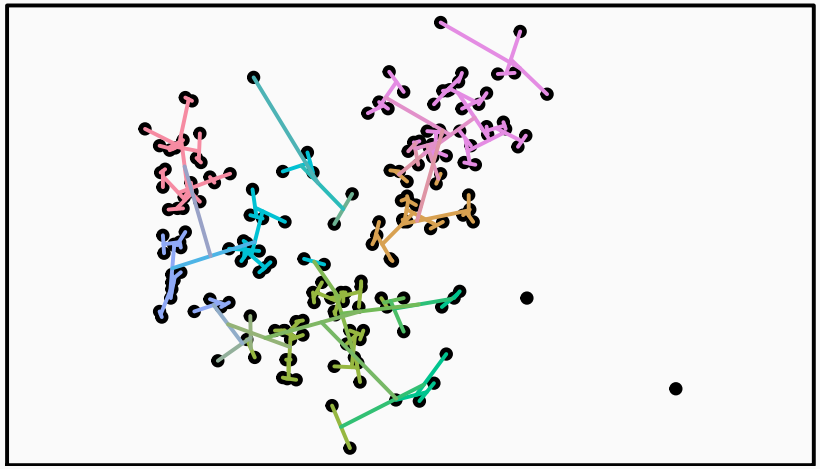
Ward



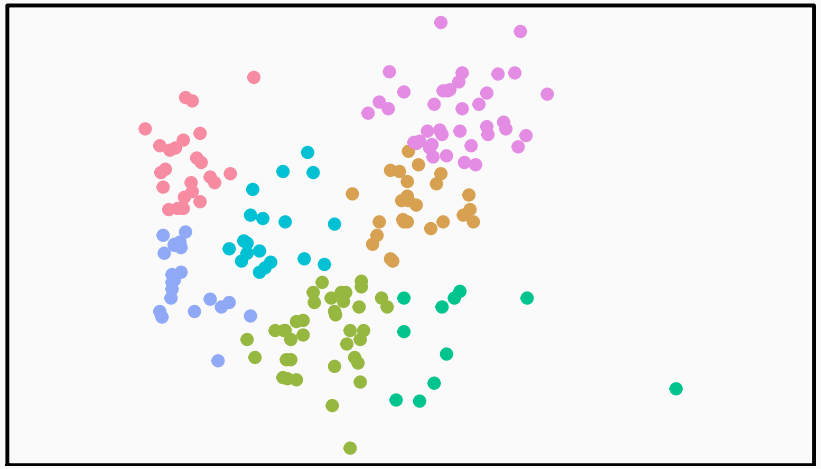
Diana



Average



K−means



Variance decomposition

- for any $C \subset \{1, \ldots, N\}$

$$\sum_{i \in C} \sum_{j \in C} \|\mathbf{x}_i - \mathbf{x}_j\|^2 = 2|C| \sum_{i \in C} \left\|\mathbf{x}_i - \widehat{\mathbf{x}}_C\right\|^2,$$

where

$$\widehat{\mathbf{x}}_C = \frac{1}{|C|} \sum_{i \in C} \mathbf{x}_i$$

- and therefore if $C_k = \{i \in \{1, \ldots, N\} \mid z_i = k\}$

$$\begin{aligned} \min_{\boldsymbol{\gamma}} \sum_{i=1}^{N} \left\|\mathbf{x}_i - \boldsymbol{\gamma}_{z_i}\right\|^2 &= \frac{1}{2} \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i \in C_k} \sum_{j \in C_k} \|\mathbf{x}_i - \mathbf{x}_j\|^2 \\ &= \frac{1}{2} W(\{C_1, \ldots, C_k\}) \end{aligned}$$

Variance and clustering

Quantization and within variance

- minimizing the quantization error is equivalent to maximizing the total within variance
- K-means is very close to Ward's method!

Total variance

- the total variance is $\sum_{i=1}^{N} \left\| \mathbf{X}_i - \widehat{\mathbf{X}}_{\mathcal{D}} \right\|^2$
- we have

$$\sum_{i=1}^{N} \left\| \mathbf{X}_i - \widehat{\mathbf{X}}_{\mathcal{D}} \right\|^2 = \min_{\mathbf{r}} \sum_{i=1}^{N} \left\| \mathbf{X}_i - \widehat{\mathbf{X}}_{C_{\gamma_{z_i}}} \right\|^2 + \sum_{k=1}^{K} |C_k| \left\| \widehat{\mathbf{X}}_k - \widehat{\mathbf{X}}_{\mathcal{D}} \right\|^2$$

$$= \frac{1}{2} W(\{C_1, \ldots, C_k\}) + \frac{1}{2} \sum_{k=1}^{K} \sum_{k'=1}^{K} |C_k||C_{k'}| \left\| \widehat{\mathbf{X}}_k - \widehat{\mathbf{X}}_{k'} \right\|^2$$

Variance and clustering

Between variance

- $B(C_1, \ldots, C_k) = \frac{1}{2} \sum_{k=1}^{K} \sum_{k'=1}^{K} |C_k||C_{k'}| \left\| \widehat{\mathbf{X}}_k - \widehat{\mathbf{X}}_{k'} \right\|^2$
- weighted pairwise distances between prototypes
- measures how spread the clusters are

Within and between variance

- total variance = within variance + between variance
- the total variance does not depend on the clustering
- by minimizing the within variance, one maximizes the between variance!
- clusters are both compact and well separated (at least at the prototype level)

Summary

K-means

- ☺ clear quantification interpretation
- ☺ compact clusters with separated prototypes
- ☺ $\Theta(NKP)$ running time
- ☺ efficient initialization strategy (k-means++)
- ☹ K?
- ☹ spherical clusters which might be very close one to another

Outline

Density based clustering

Cluster = dense region

- clusters are areas of high density compared to other areas
- density based separation: clusters are separated by low density areas
- no direct assumption on cluster shape and on relative distances

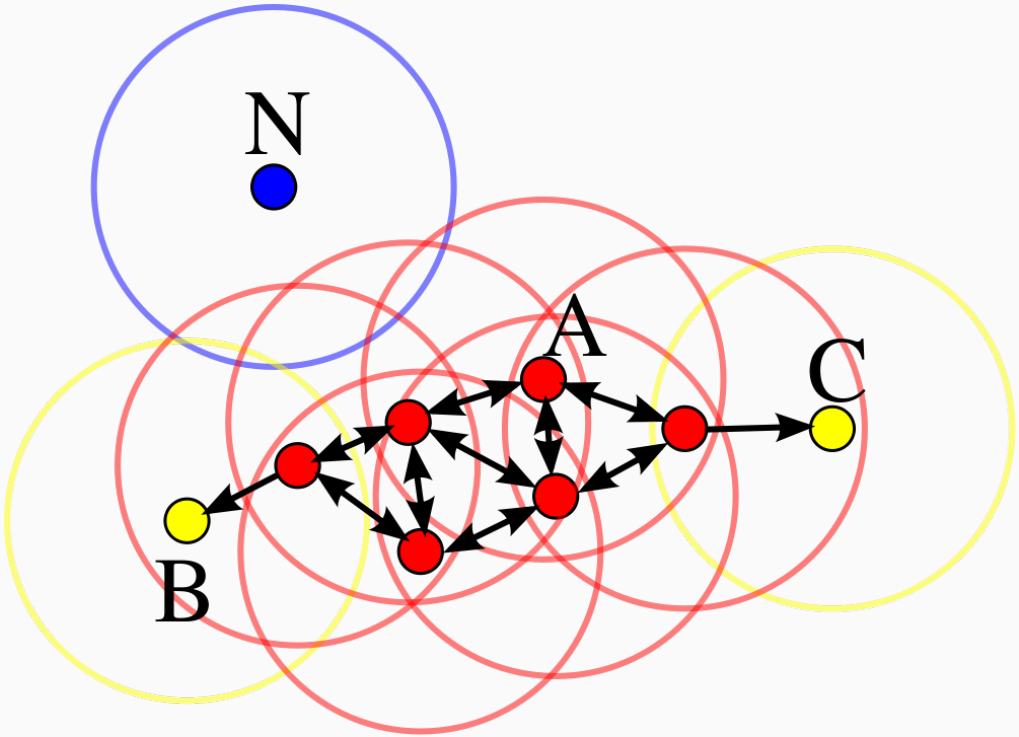
DBSCAN

- Ester, Kriegel, Sander & Xu, 1996 [2]
- most well known density based algorithm
- simple density model:
  - enough points in a tight region
  - two parameters:
    - minPts: minimal number of points in a region
    - ε: radius of the region

Several types of points

- *core point*: $\mathbf{X}_i$ such that $|\{\mathbf{X} \in \mathcal{D} | d(\mathbf{X}_i, \mathbf{X}) \leq \varepsilon\}| \geq \mathit{minPts}$
- *directly density reachable point* from a core point $\mathbf{X}_i$: $\mathbf{X}_j$ such that $d(\mathbf{X}_i, \mathbf{X}_j) \leq \varepsilon$
- *density reachable point* from a core point $\mathbf{X}_i$: $\mathbf{X}_j$ such that there is a chain of core points $\mathbf{X}_{k_1}, \ldots, \mathbf{X}_{k_l}$ with $d(\mathbf{X}_{k_t}, \mathbf{X}_{k_{t+1}}) \leq \varepsilon$, $\mathbf{X}_{k_1} = \mathbf{X}_j$ and $\mathbf{X}_{k_l} = \mathbf{X}_i$
- *border point*: *density reachable points* that are not core points
- *density connected points*: two points are density connected is they are density reachable from the same (core) point
- *noise point*: points that are not density reachable from a core point

Types of points



- A and red points: core points
- B and C: border points
- N: noise point

illustration from https://commons.wikimedia.org/wiki/File:DBSCAN-Illustration.svg

Clusters and noise

DBSCAN clusters

A cluster in DBSCAN is a maximal set of density connected points.

Noise

Points that do not belong to DBSCAN clusters form the noise.

Clustering model

- DBSCAN produces a partition of $\mathcal{D}$ into $C_1, \ldots, C_K, N$, where the C_k are the clusters and N is the noise
- K is not specified directly but only a consequence of minPts and ε
- notice that $K \leq \frac{N}{minPts}$ as a cluster must contain a core point and has therefore a minimal size of minPts
- border points can belong to several clusters and a tie breaking criterion is used to assign them to a single one

Algorithm

for all X $\in \mathcal{D}$ **do**
    if X is not labelled **then**
        $N \leftarrow \varepsilon$-neighborhood of **X**
        if $|N| <$ *minPts* **then**
            label **X** as noise
        else
            label **X** with a new cluster label k
            label with k all density reachable points from **X** (including noise ones)
        end if
    end if
end for

Complexity

- core operation: ε-neighborhood calculation
- done once for each point in $\mathcal{D}$
- naive complexity in $N \Rightarrow \Theta\left(N^2\right)$
- spatial indexing (R* tree, for instance) might decrease the cost but **not to** $\Theta\left(() N \log N\right)$
- minimal cost: $\Theta\left(N^{\frac{4}{3}}\right)$

Parameters

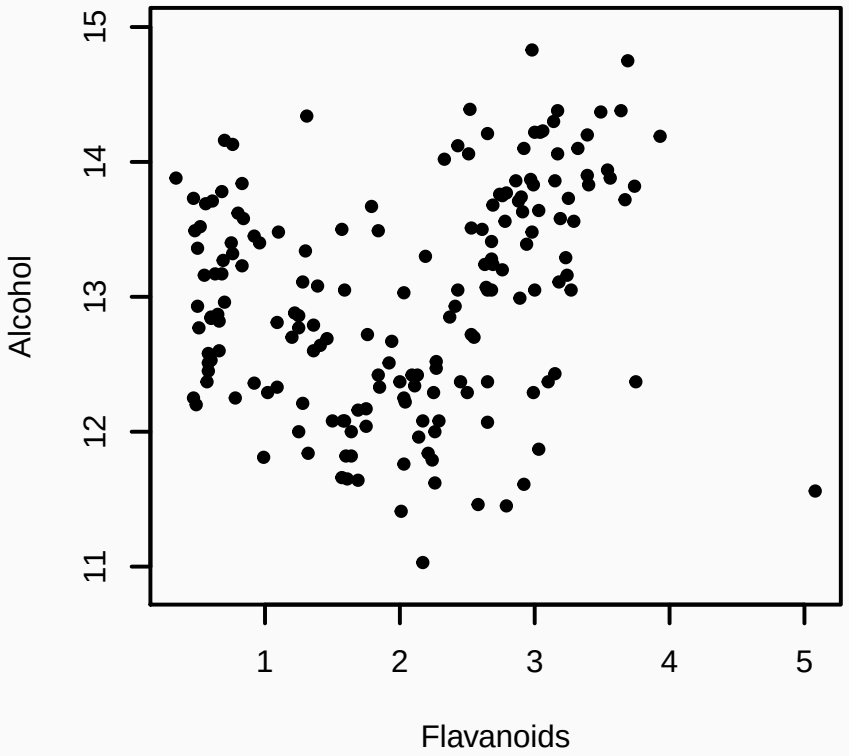
minPts

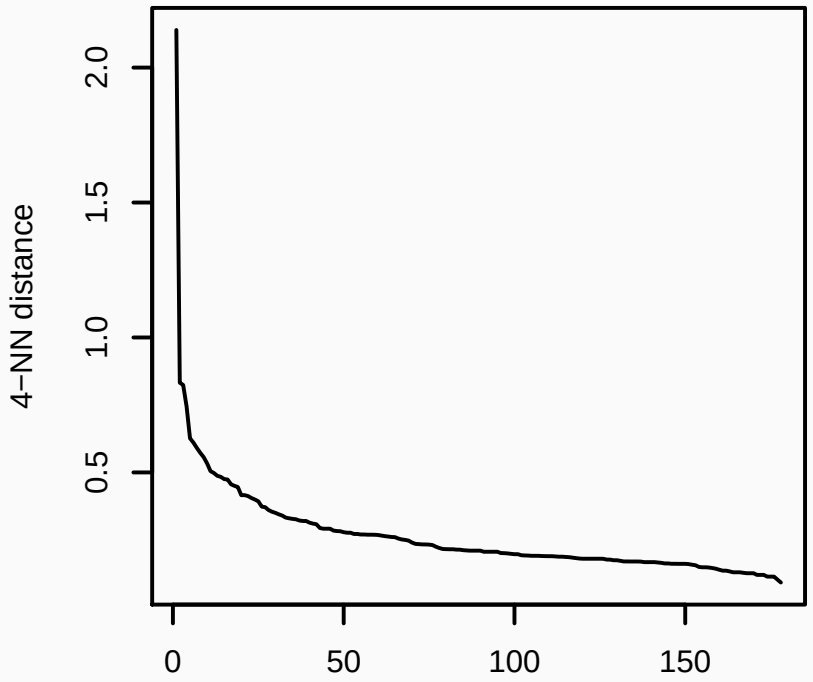
- limited impact on the results above a minimal value
- original recommendation: $minPts = 4$
- current recommendation: $minPts = 2 \times P$ (larger values for noisy data)

ε

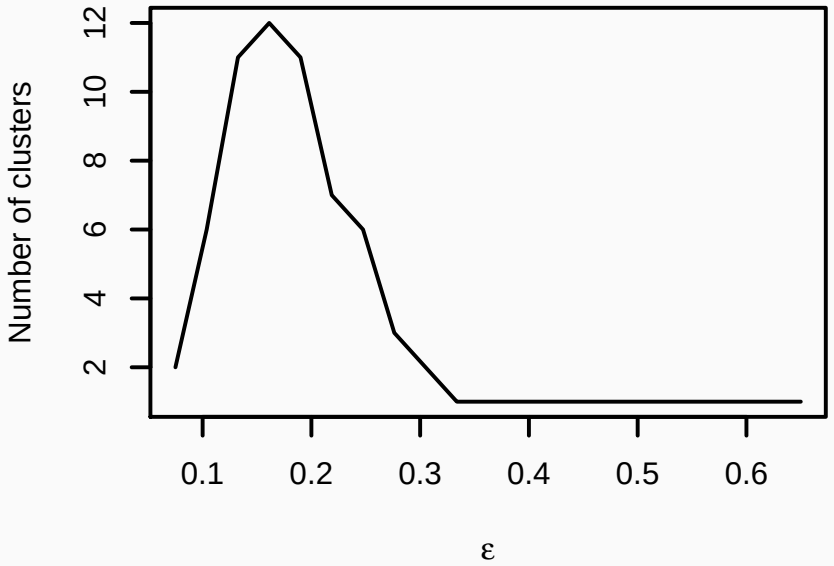
- very difficult to set
- plays a role similar to the one of k in the k-means
- one should explore the effects of using different values of ε
- "elbow" approach on the k-nn distance graph

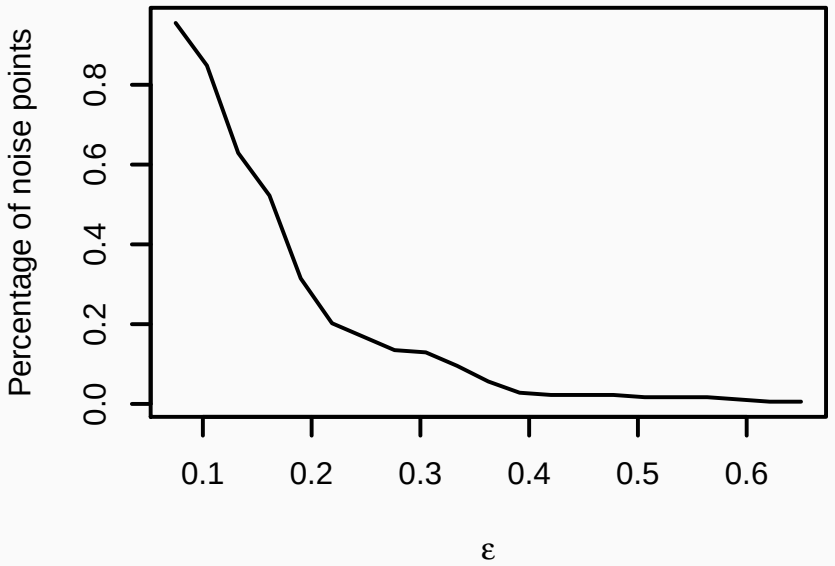
Example

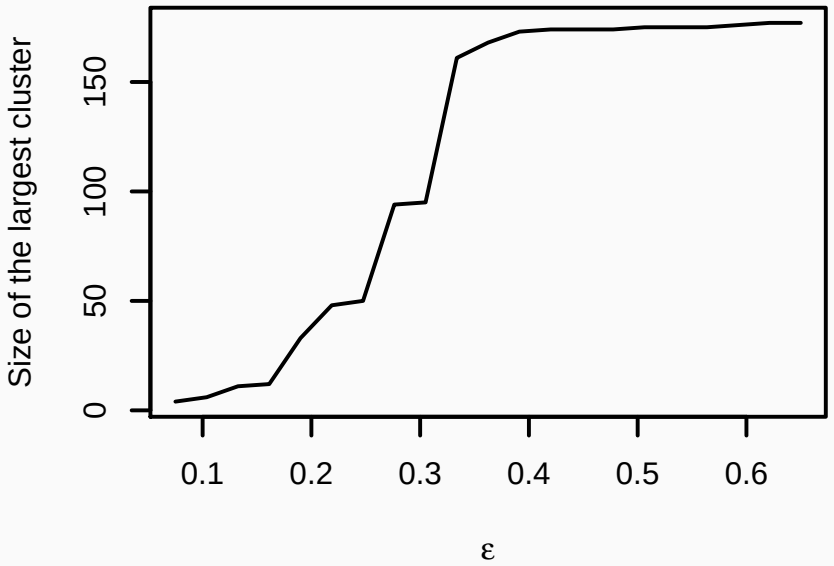

Alcohol
11
12
13
14
15
1
2
3
4
5
Flavanoids


4−NN distance
0.5
1.0
1.5
2.0
0
50
100
150

Diagnostic plots

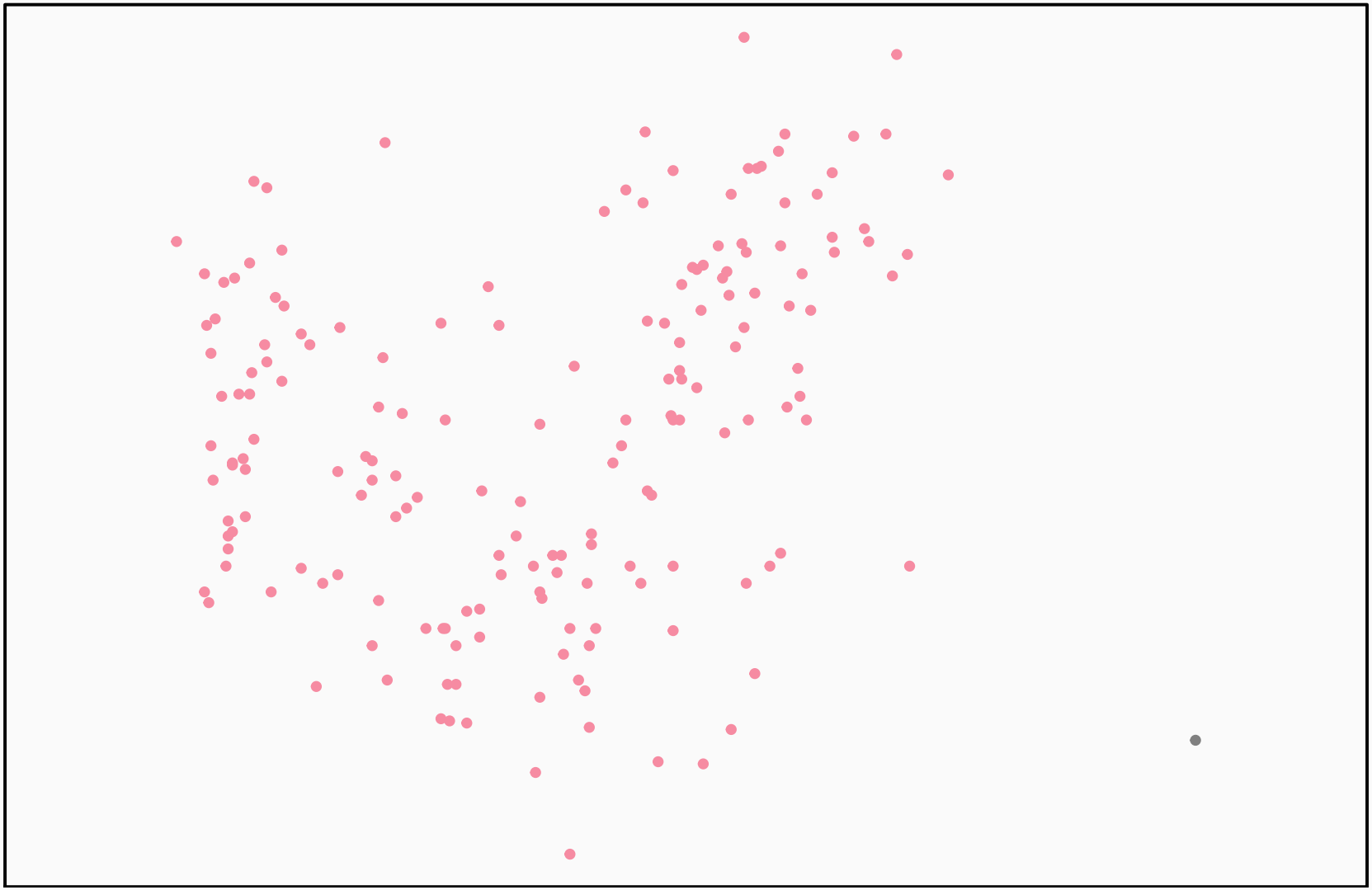






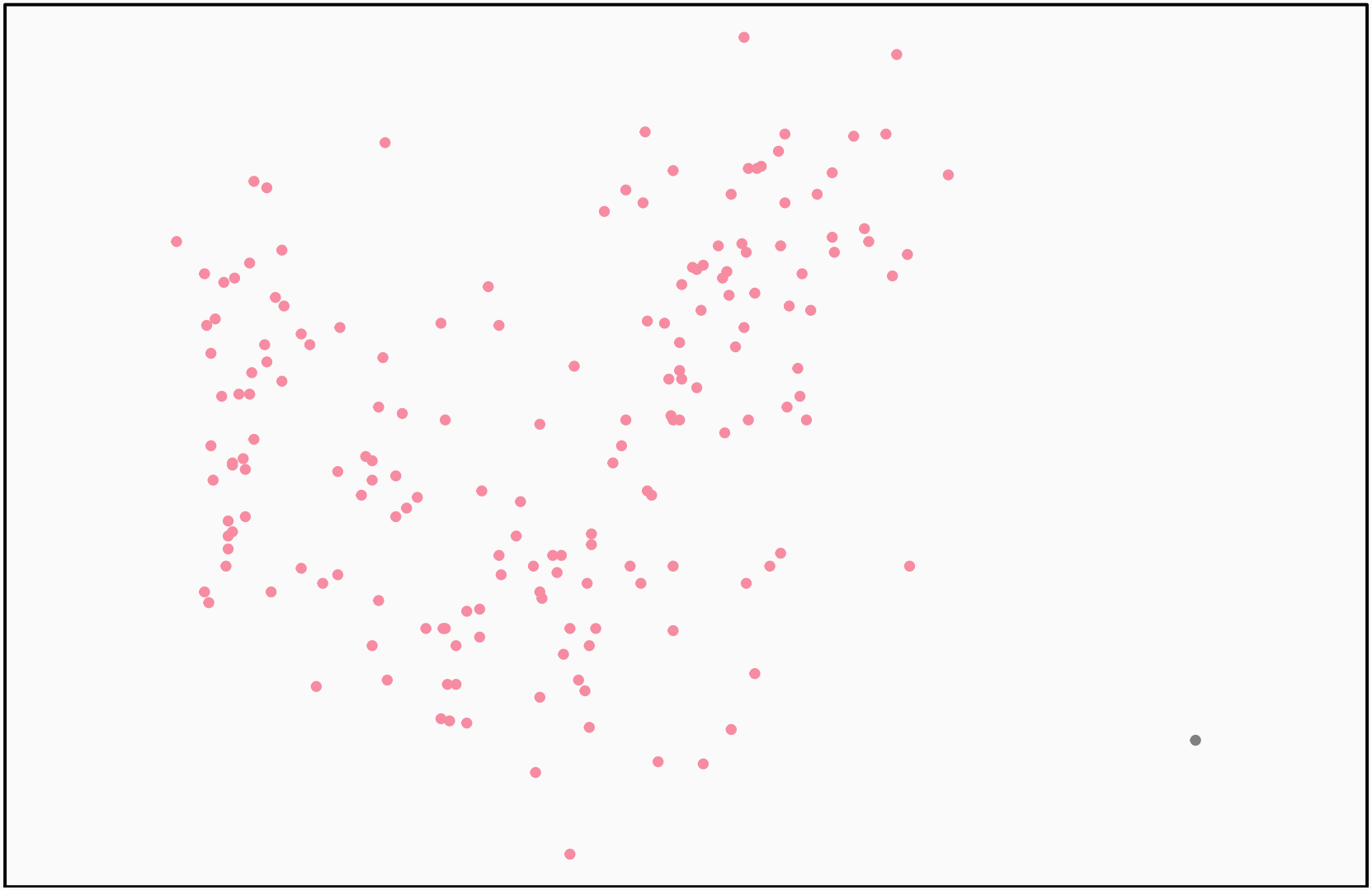
Clusters

$\varepsilon = 0.65 \quad k = 1 \text{ noise} = 1$



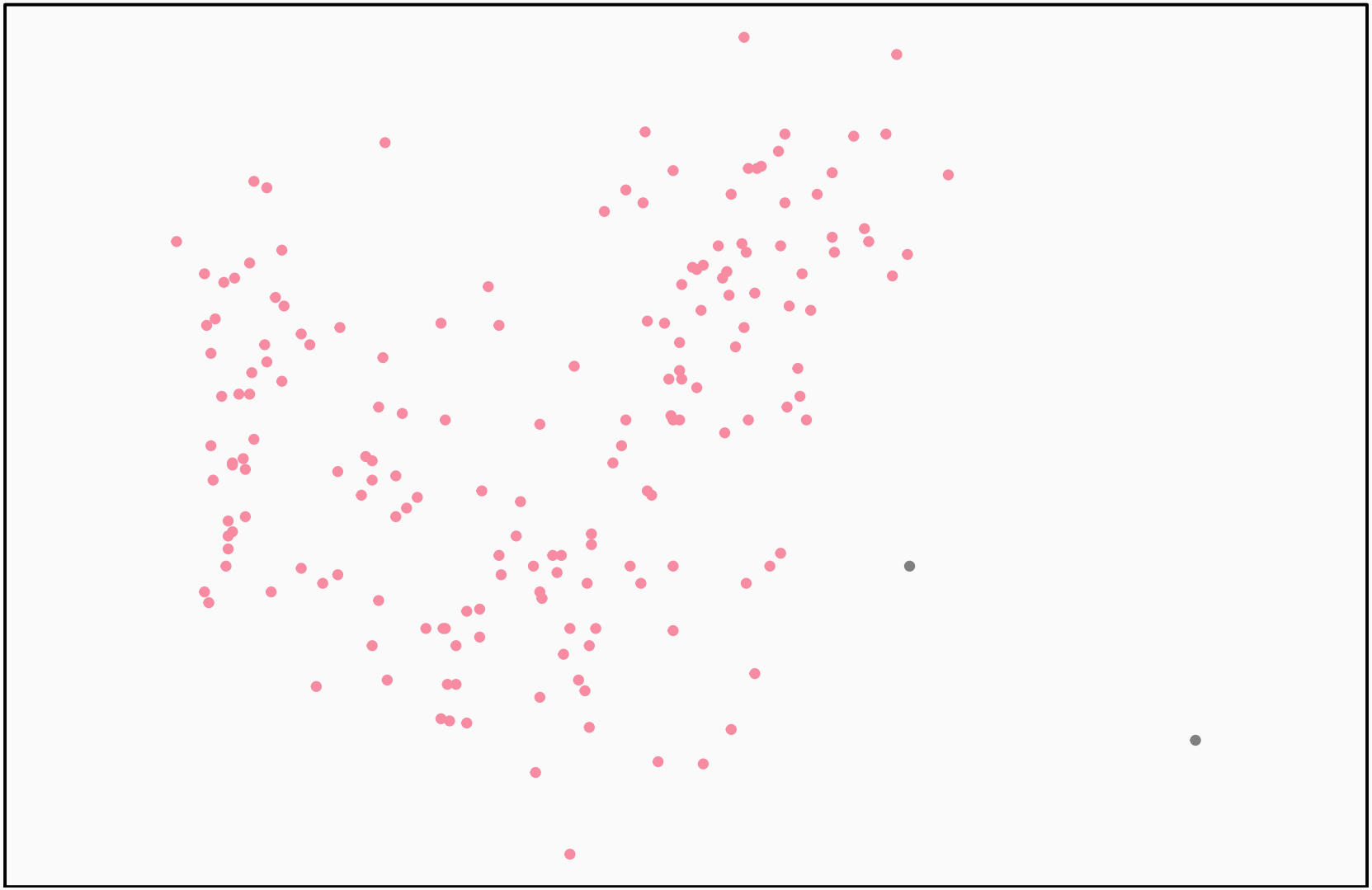
Clusters

$\varepsilon = 0.62125 \quad k = 1 \text{ noise} = 1$



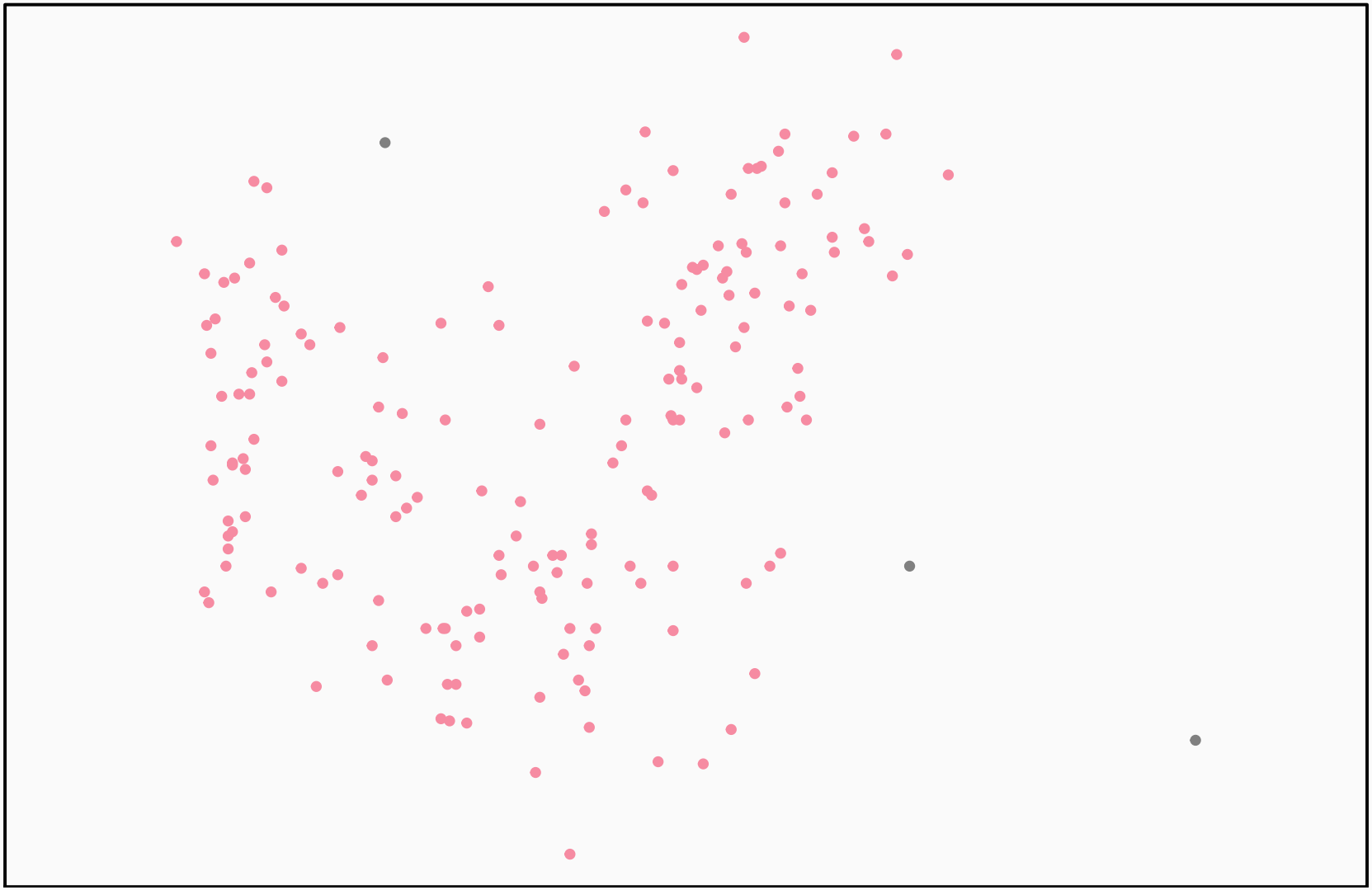
Clusters

$\varepsilon = 0.5925 \quad k = 1 \text{ noise} = 2$



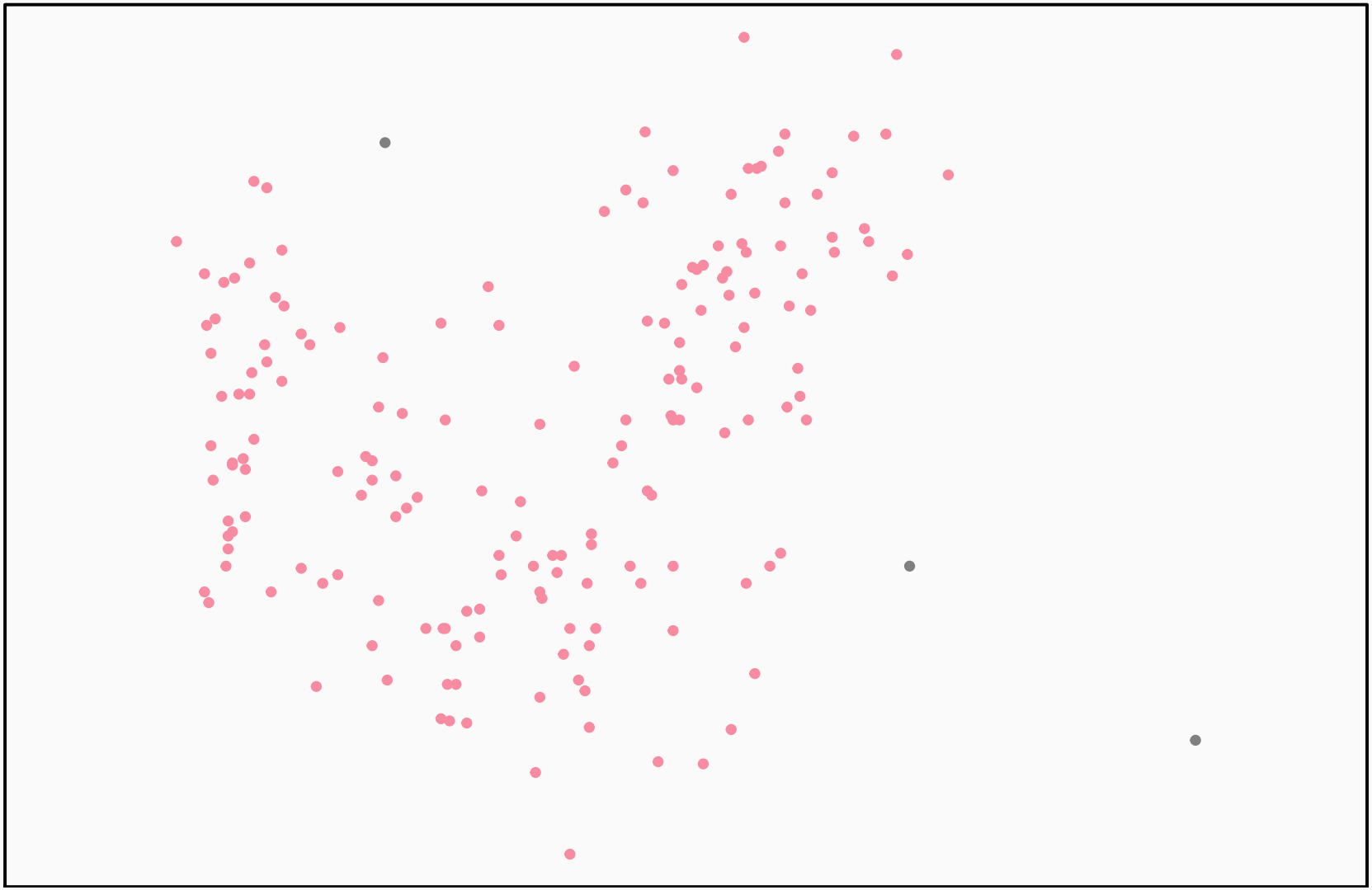
Clusters

$\varepsilon = 0.56375 \quad k = 1 \text{ noise} = 3$



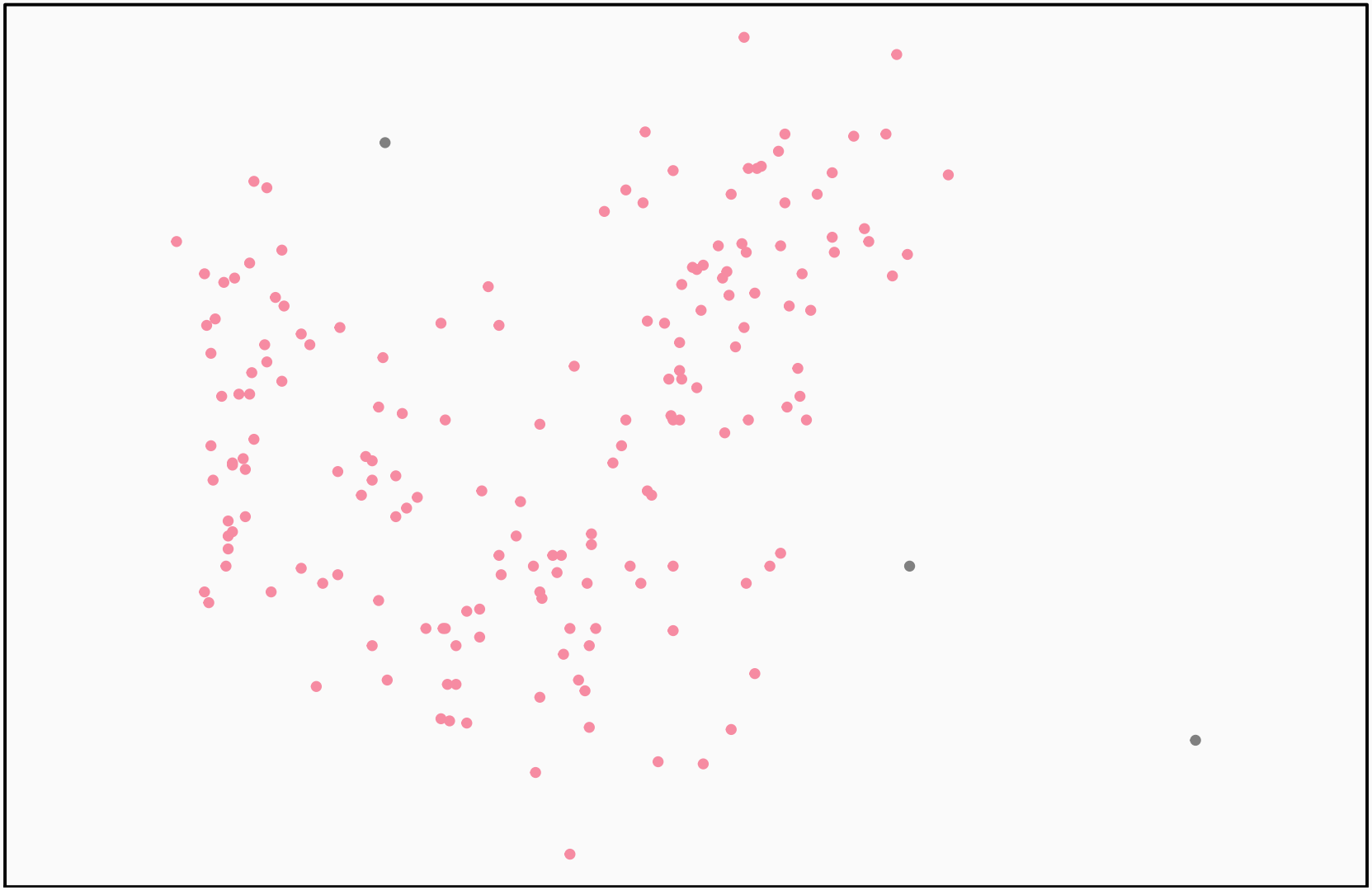
Clusters

$\varepsilon = 0.535 \quad k = 1 \text{ noise} = 3$



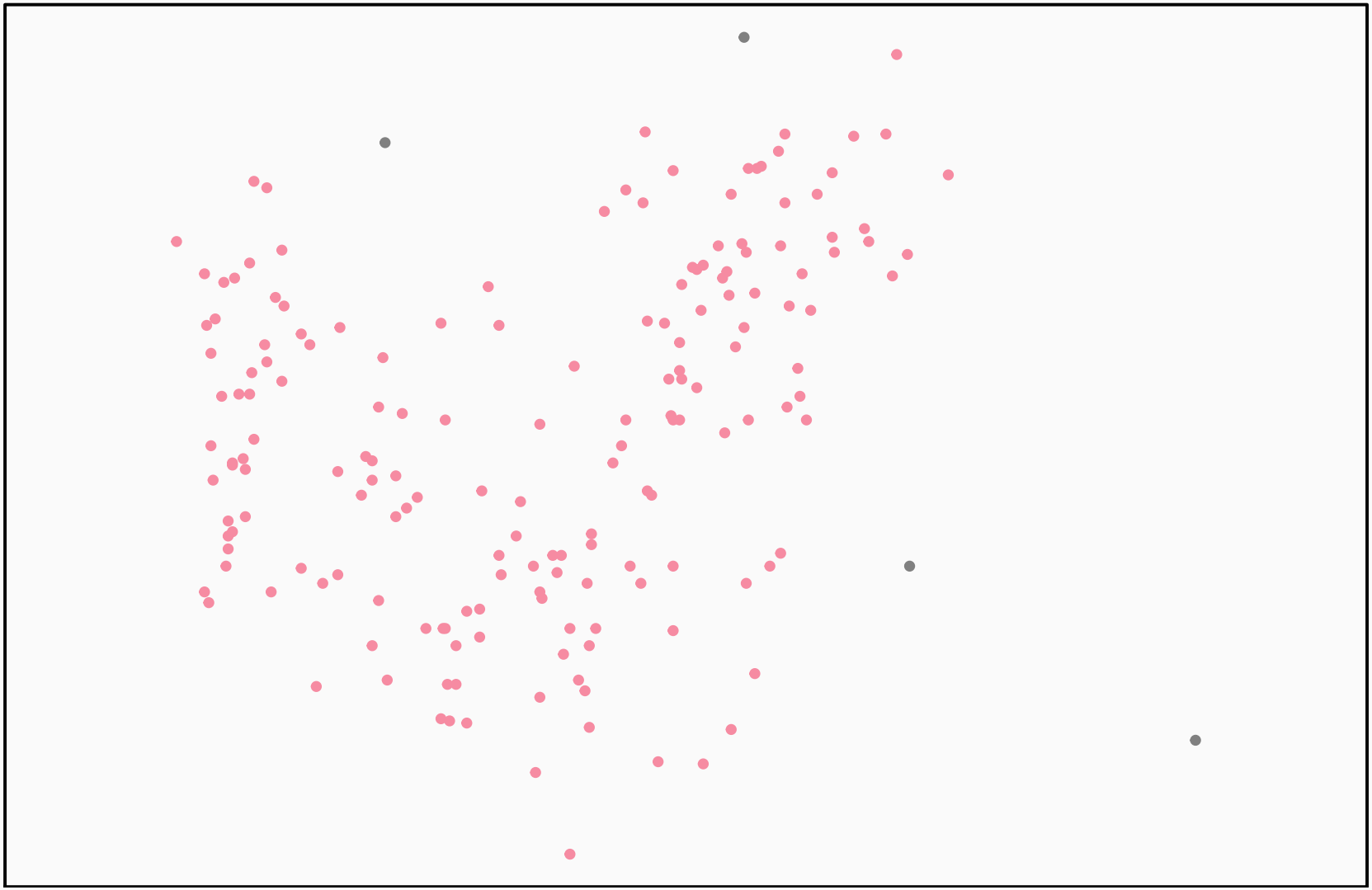
Clusters

$\varepsilon = 0.50625 \quad k = 1 \text{ noise} = 3$



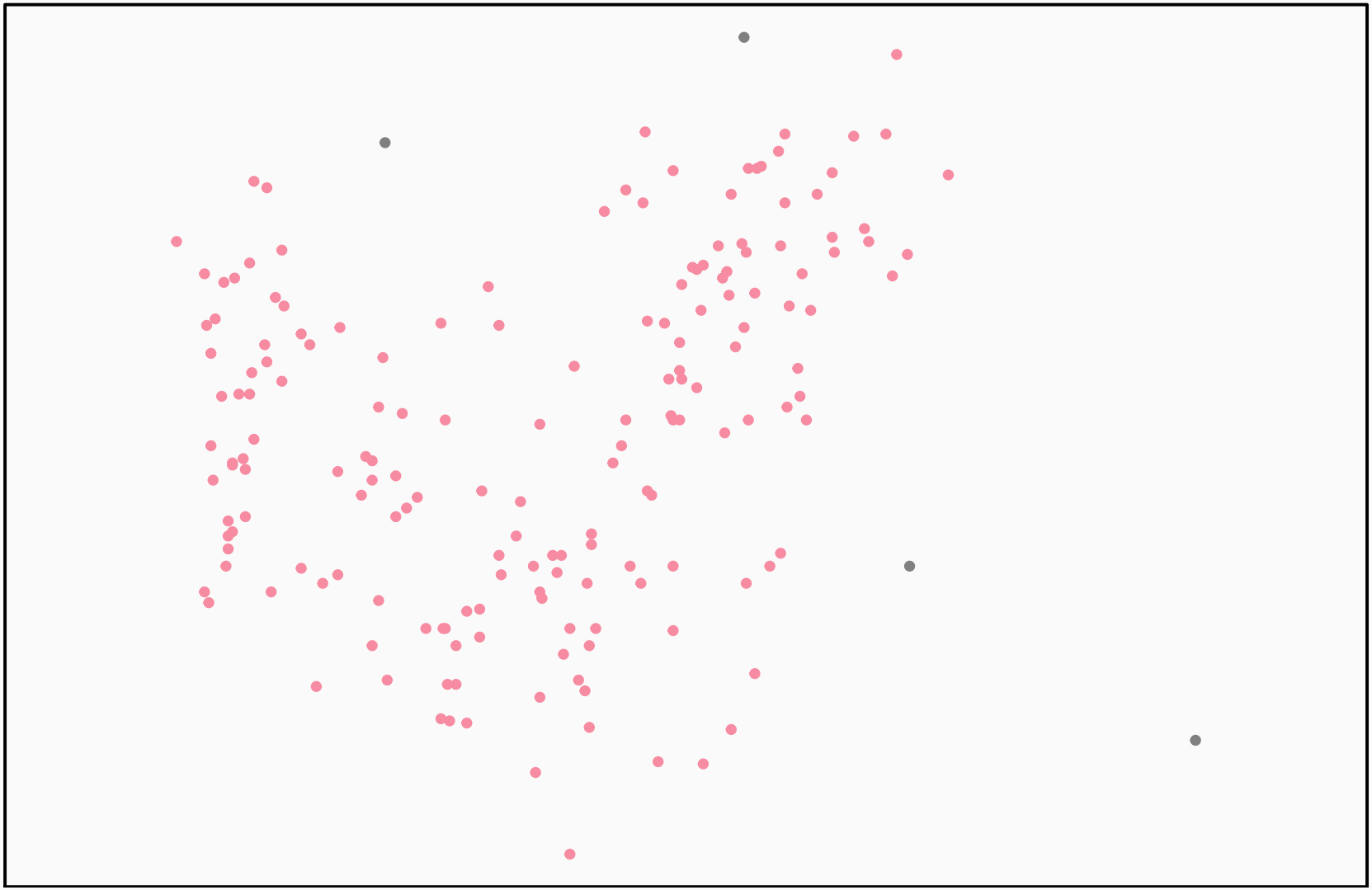
Clusters

$\varepsilon = 0.4775 \quad k = 1 \text{ noise} = 4$



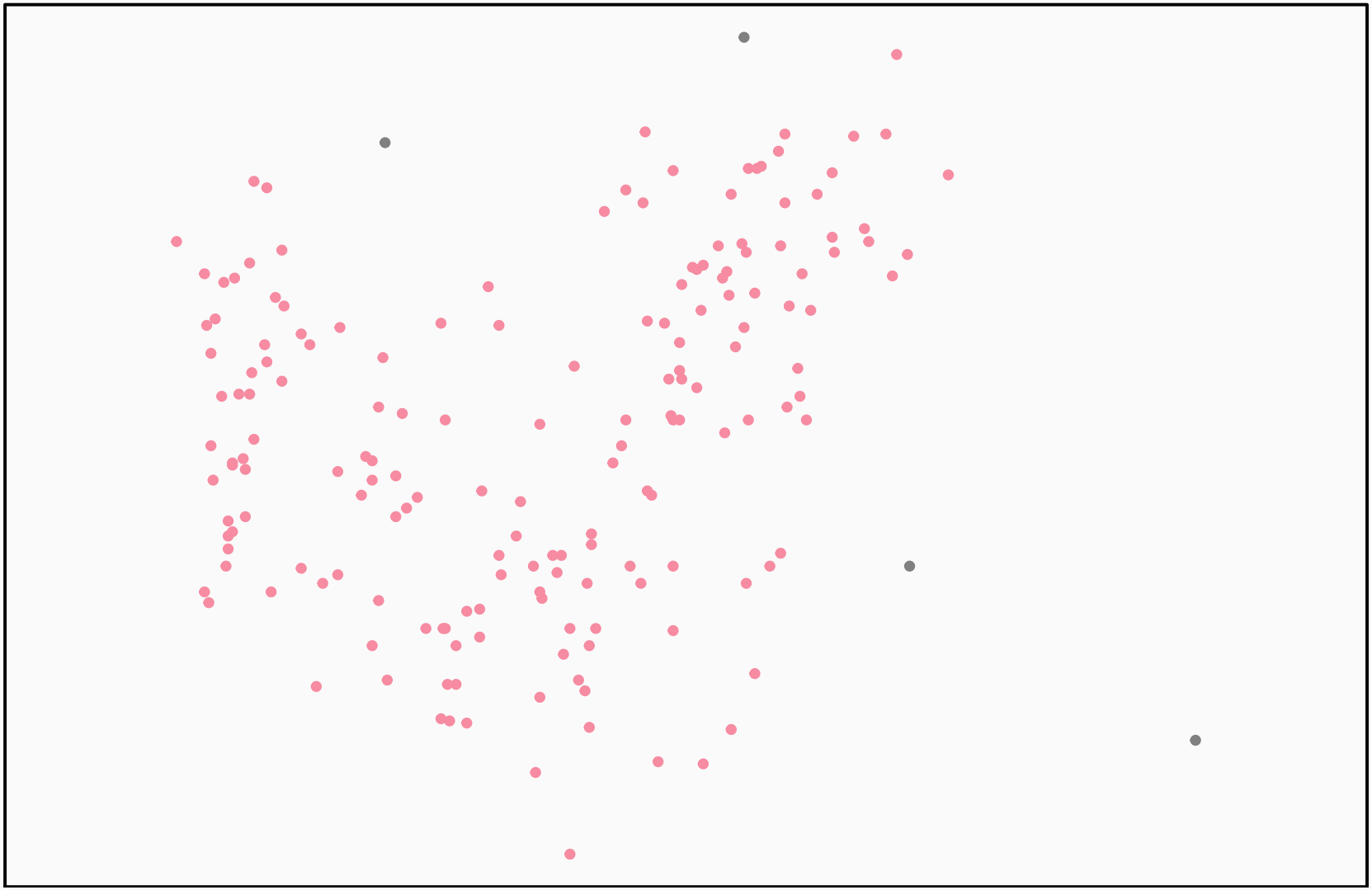
Clusters

$\varepsilon = 0.44875 \quad k = 1 \text{ noise} = 4$



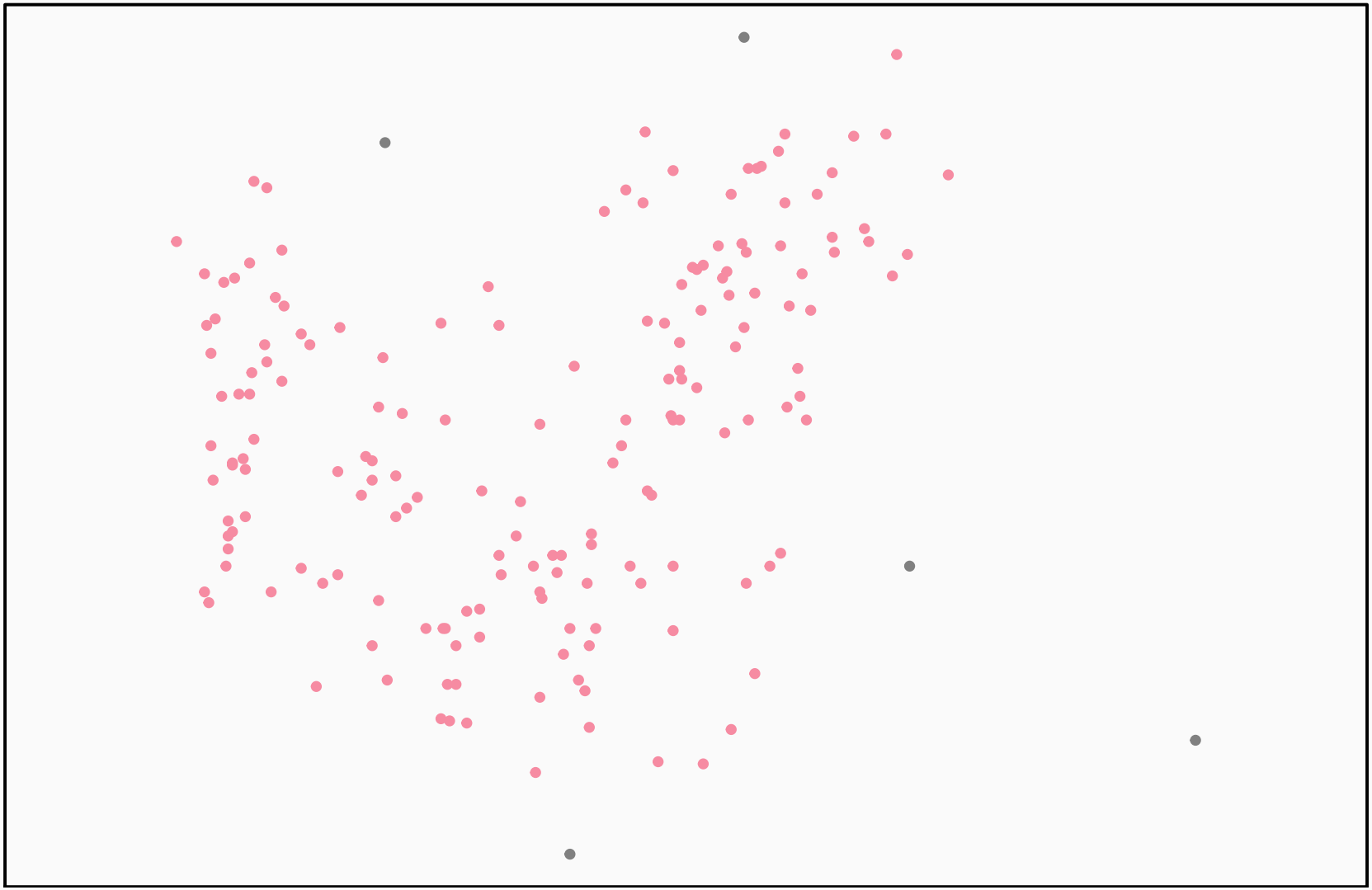
Clusters

$\varepsilon = 0.42 \quad k = 1 \text{ noise} = 4$



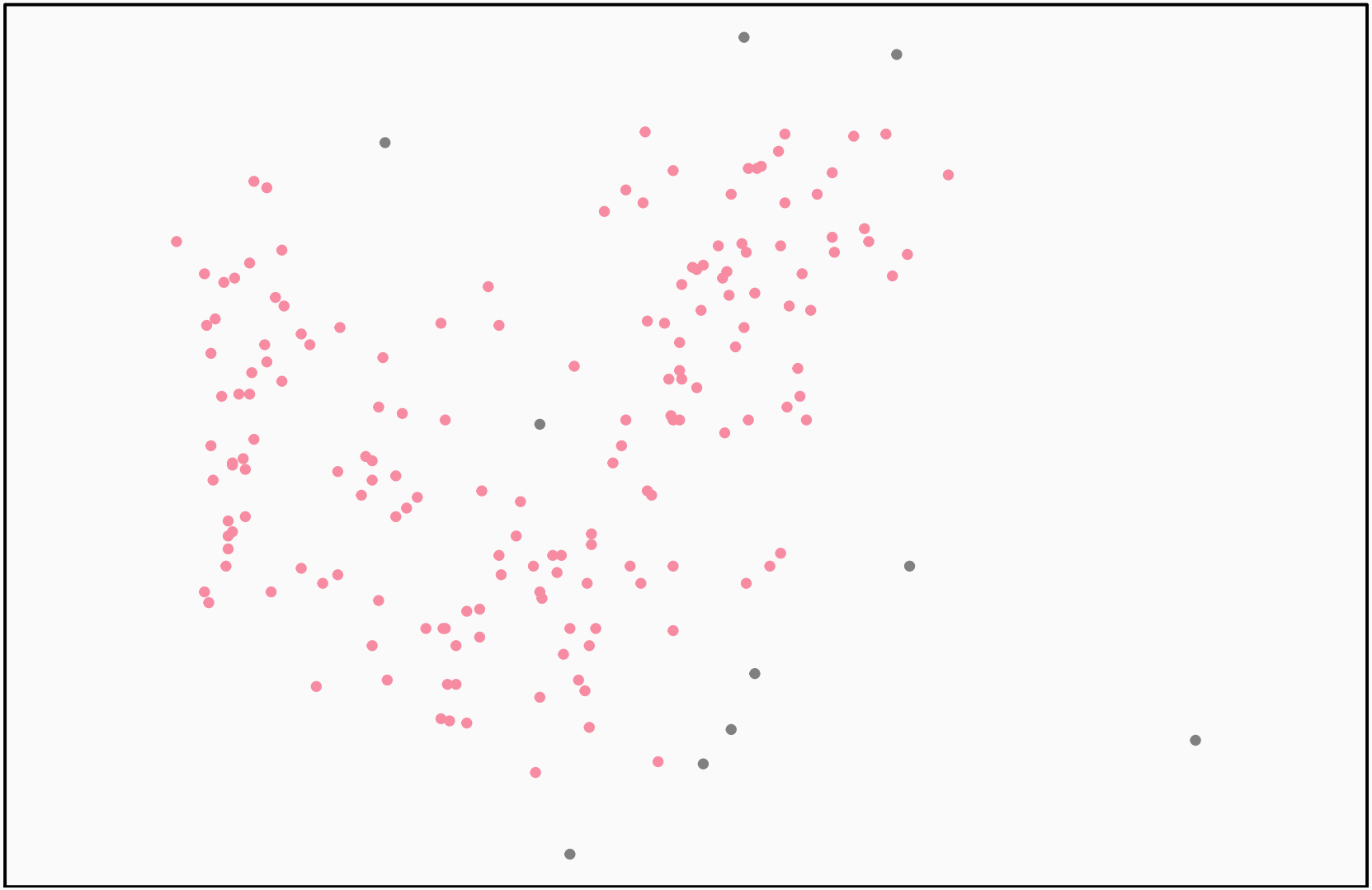
Clusters

$\varepsilon = 0.39125 \quad k = 1 \text{ noise} = 5$



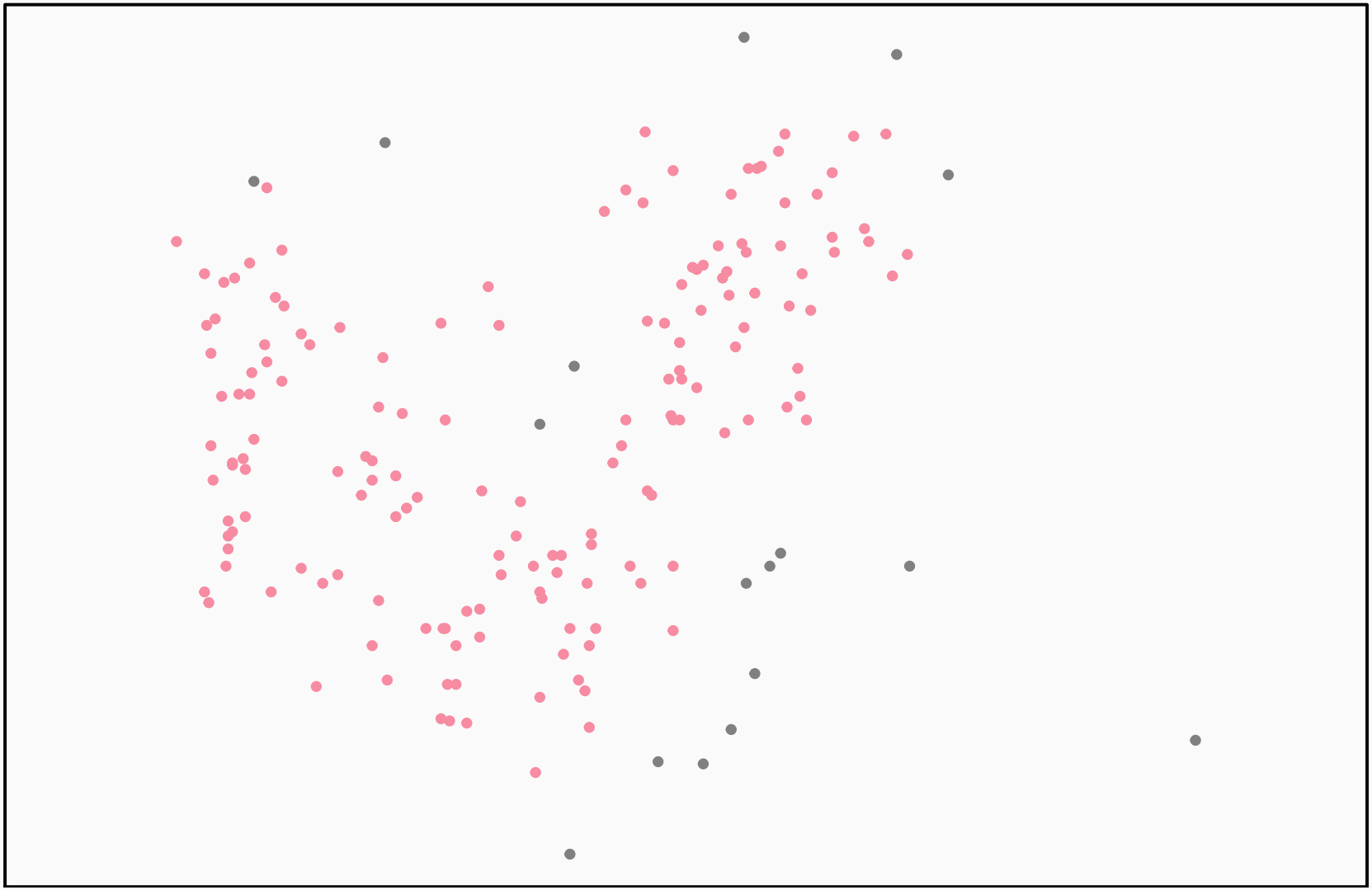
Clusters

$\varepsilon = 0.3625 \quad k = 1 \text{ noise} = 10$



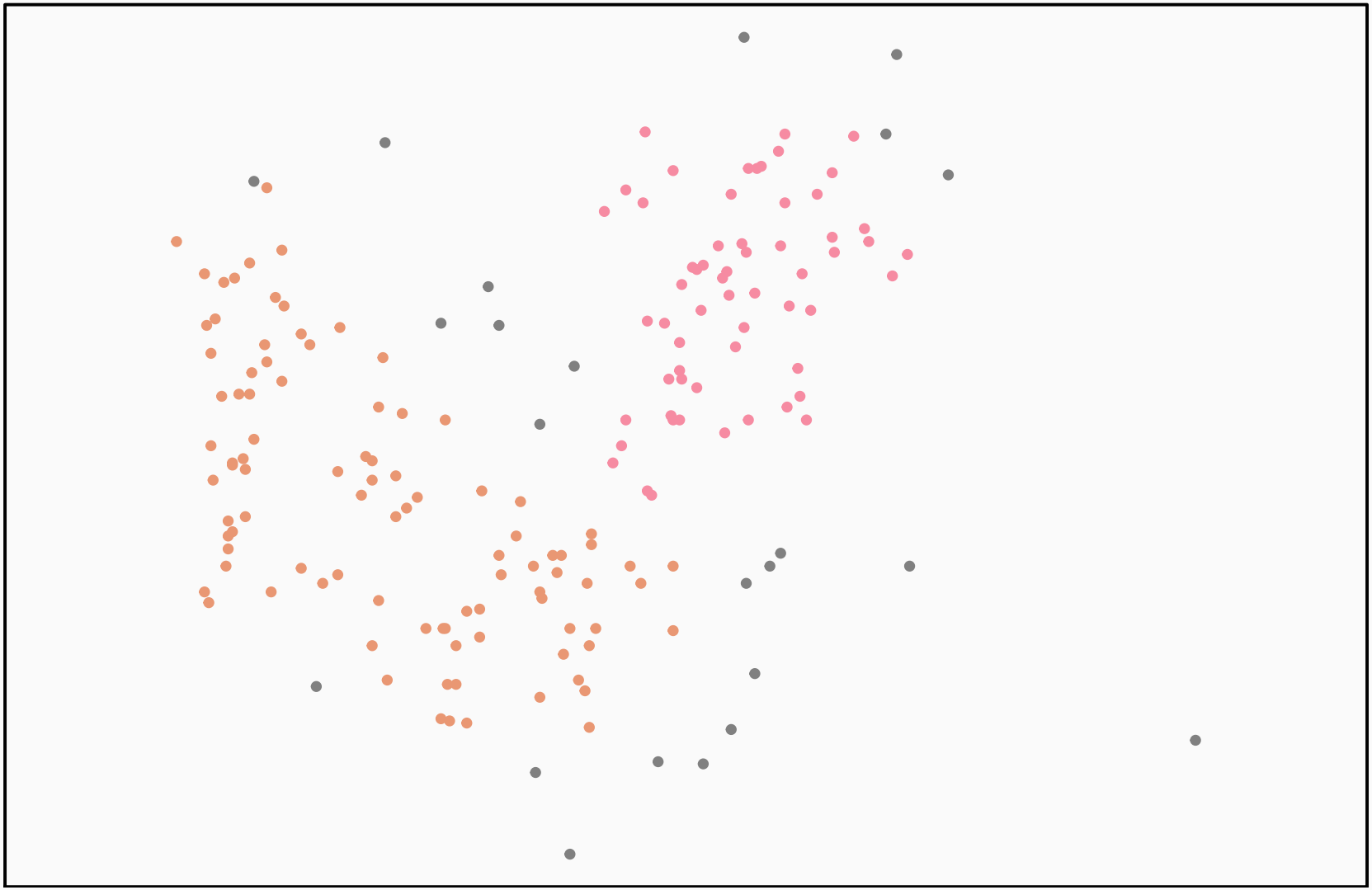
Clusters

$\varepsilon = 0.33375 \quad k = 1 \text{ noise} = 17$



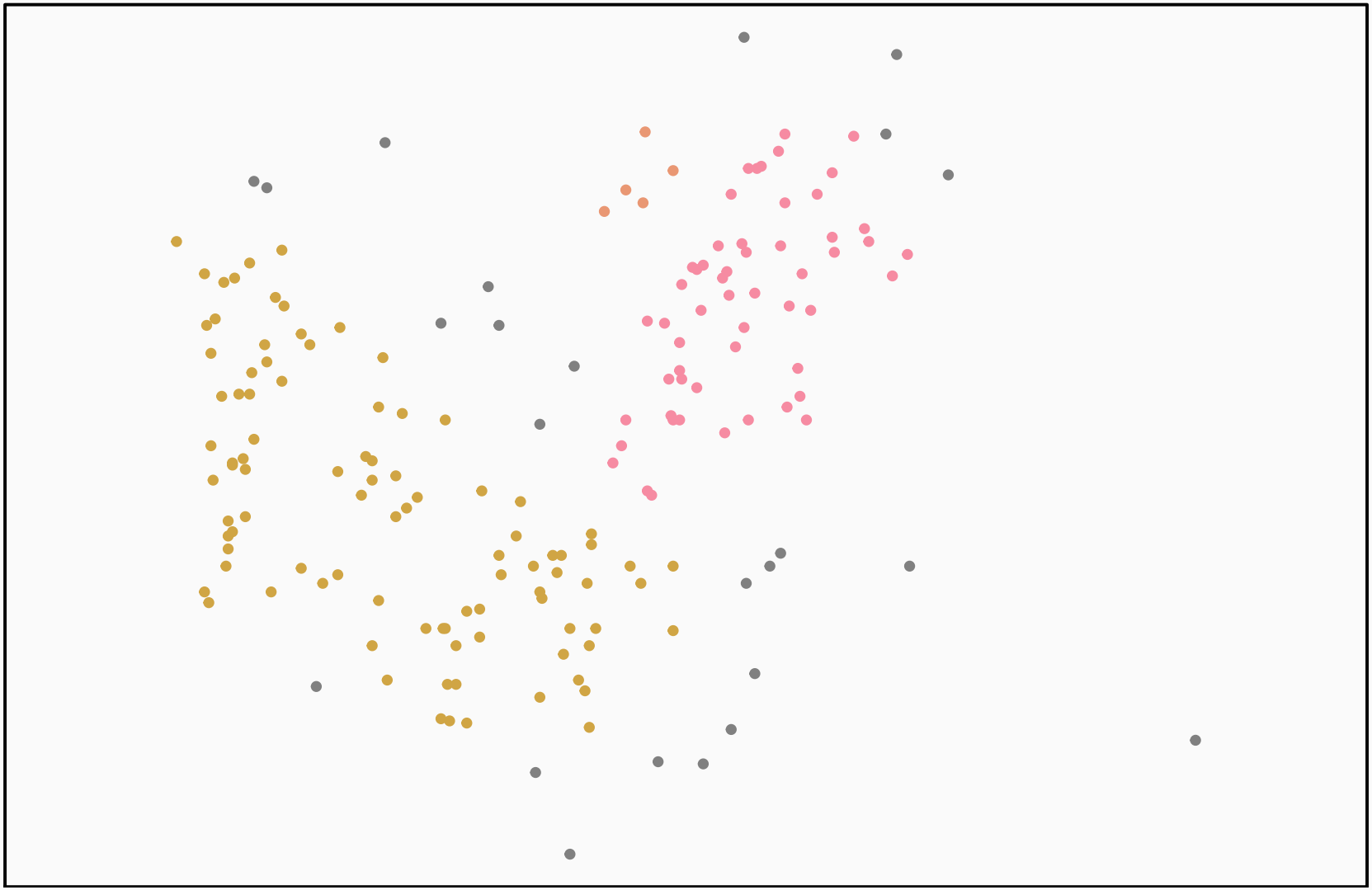
Clusters

$\varepsilon = 0.305 \quad k = 2 \text{ noise} = 23$



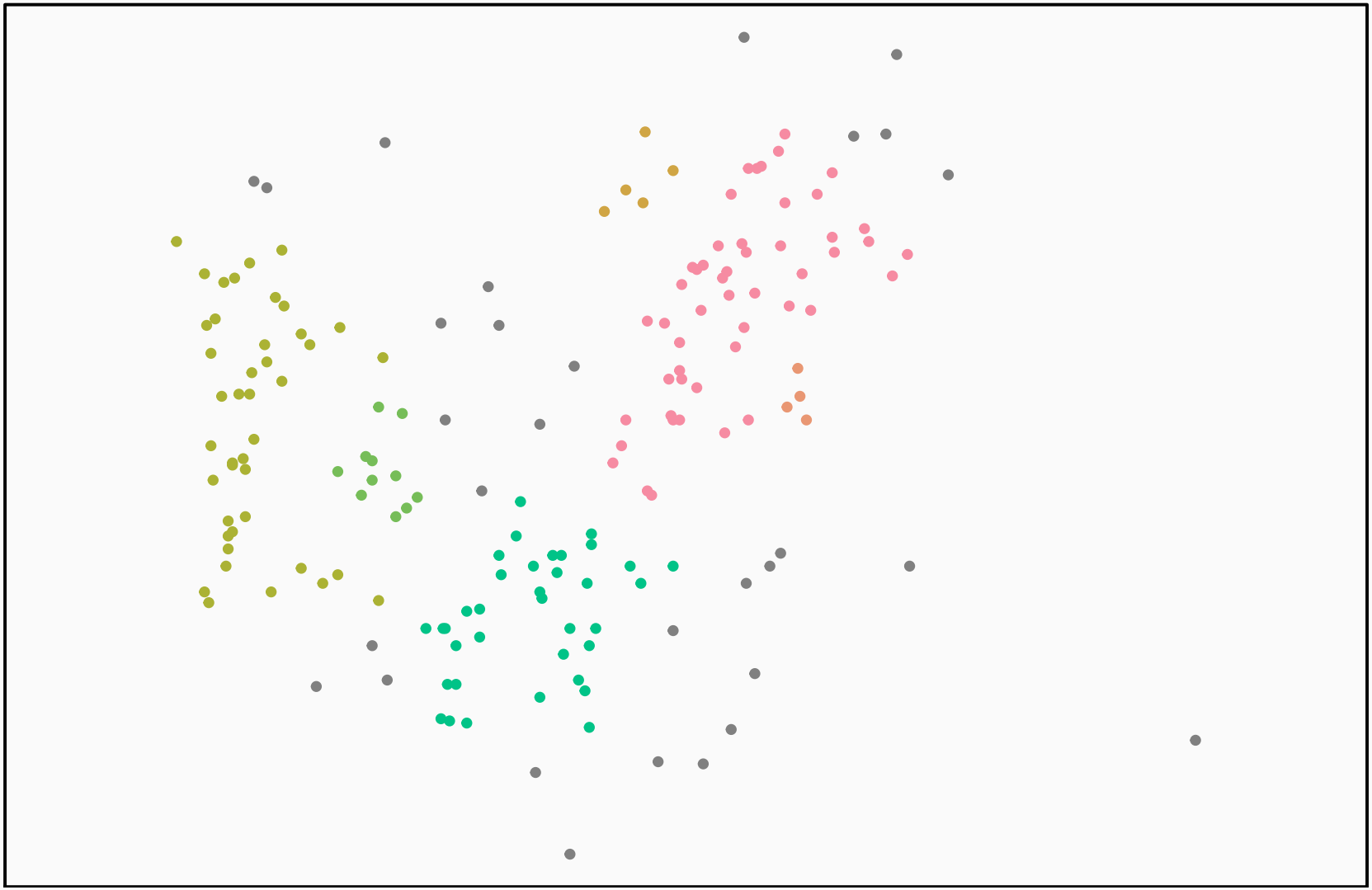
Clusters

$\varepsilon = 0.27625 \quad k = 3 \text{ noise} = 24$



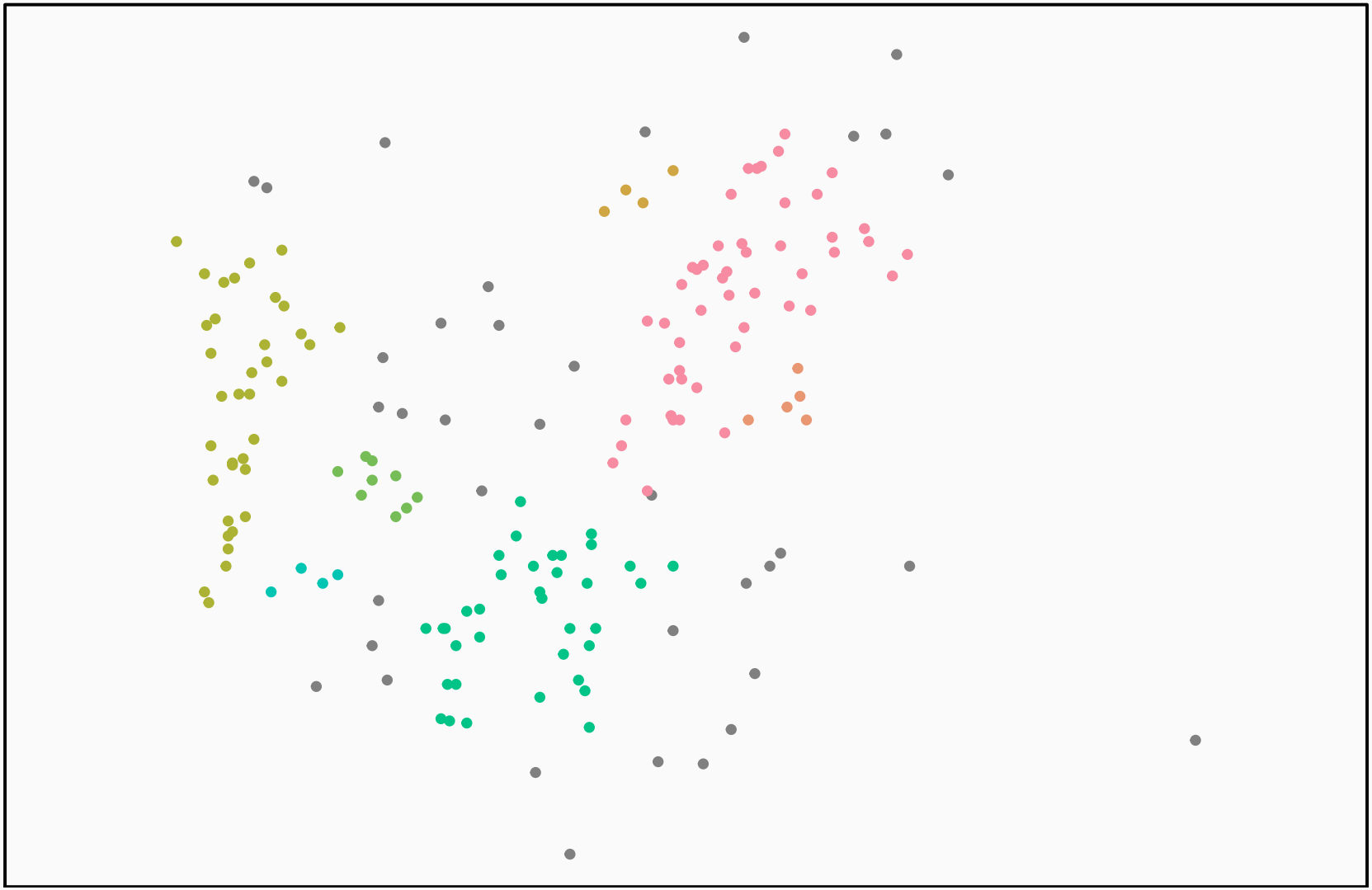
Clusters

$\varepsilon = 0.2475 \quad k = 6 \text{ noise} = 30$



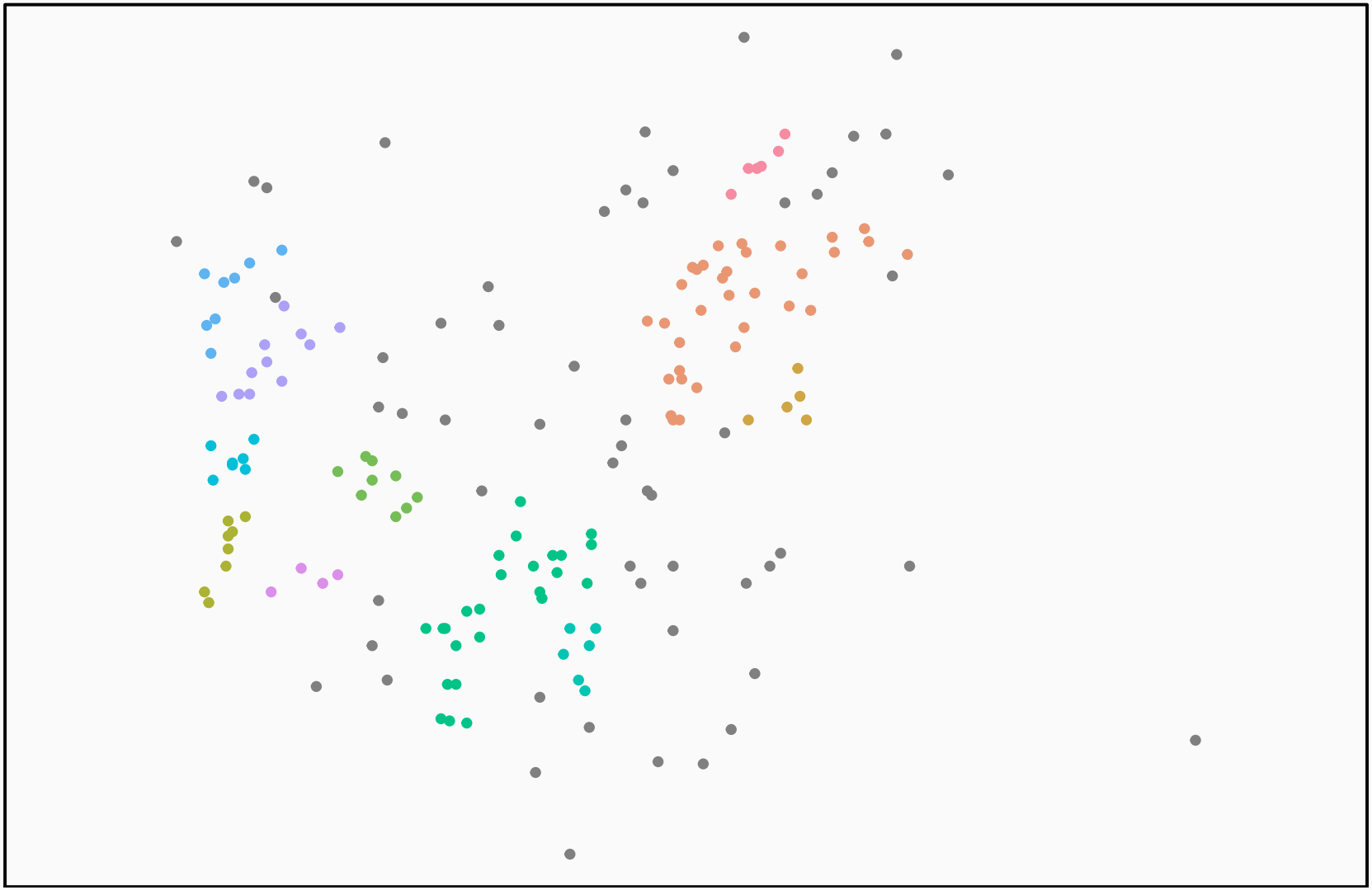
Clusters

$\varepsilon = 0.21875 \quad k = 7 \text{ noise} = 36$



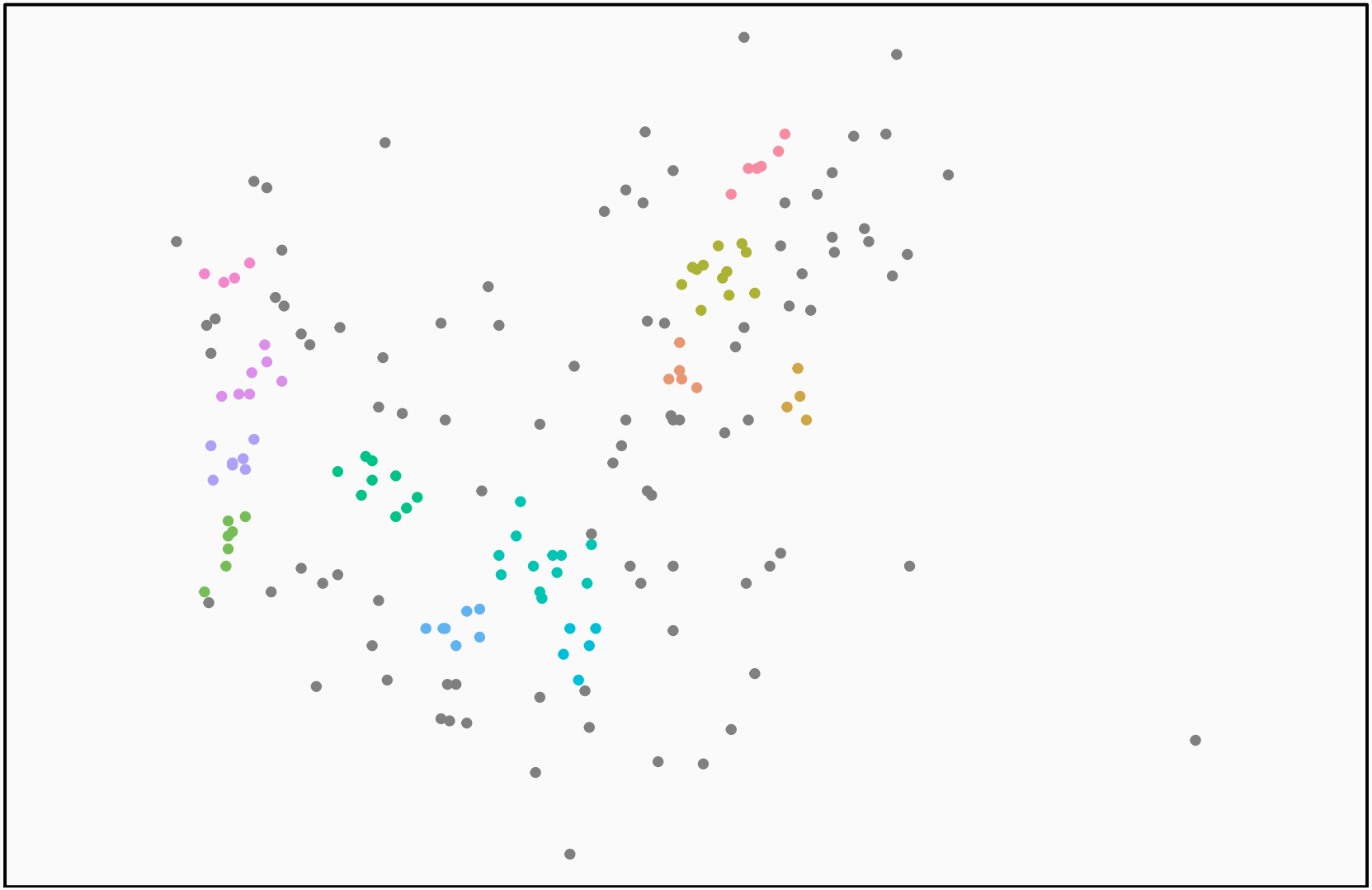
Clusters

$\varepsilon = 0.19 \quad k = 11 \text{ noise} = 56$



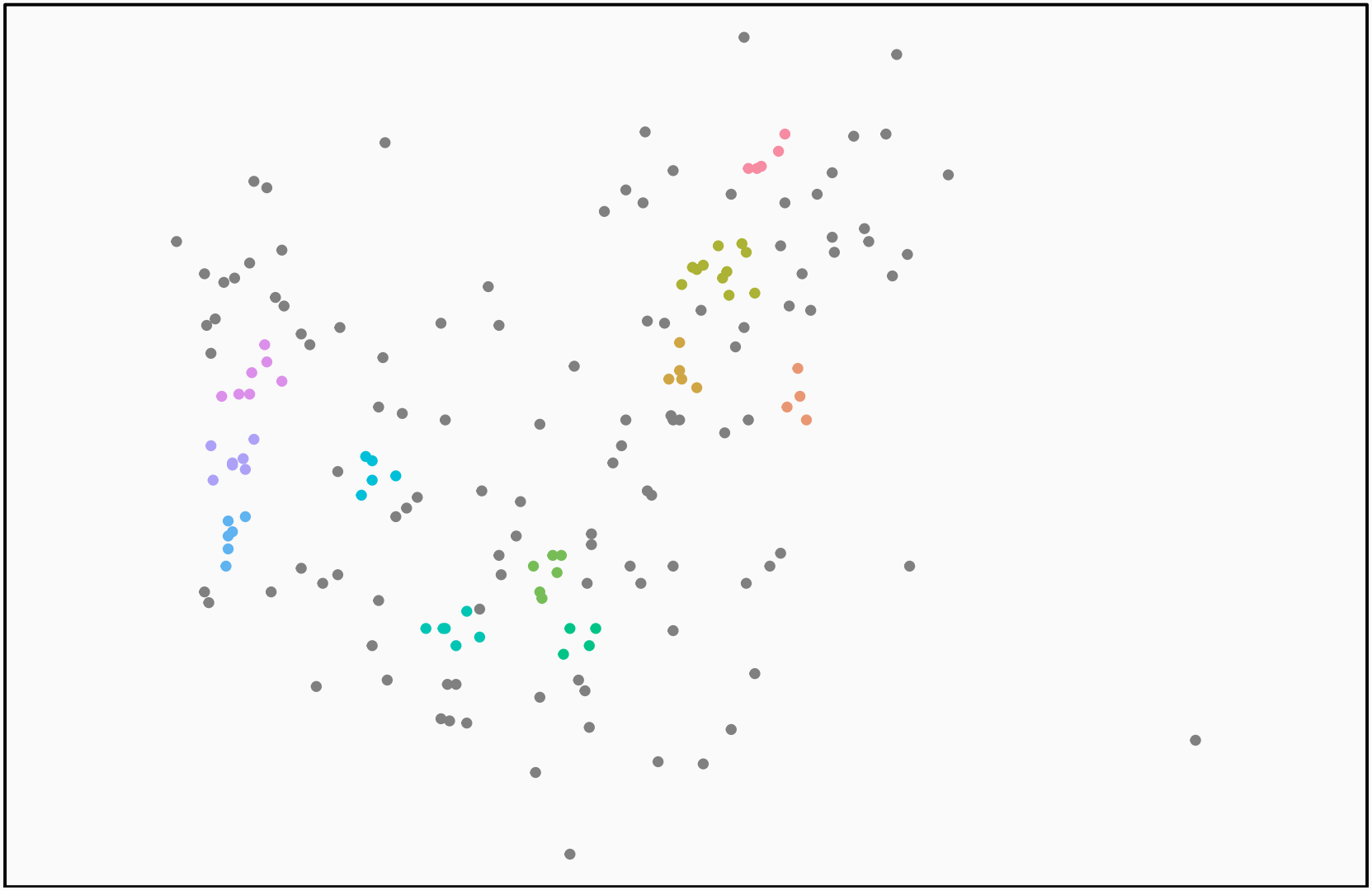
Clusters

$\varepsilon = 0.16125 \quad k = 12 \text{ noise} = 93$



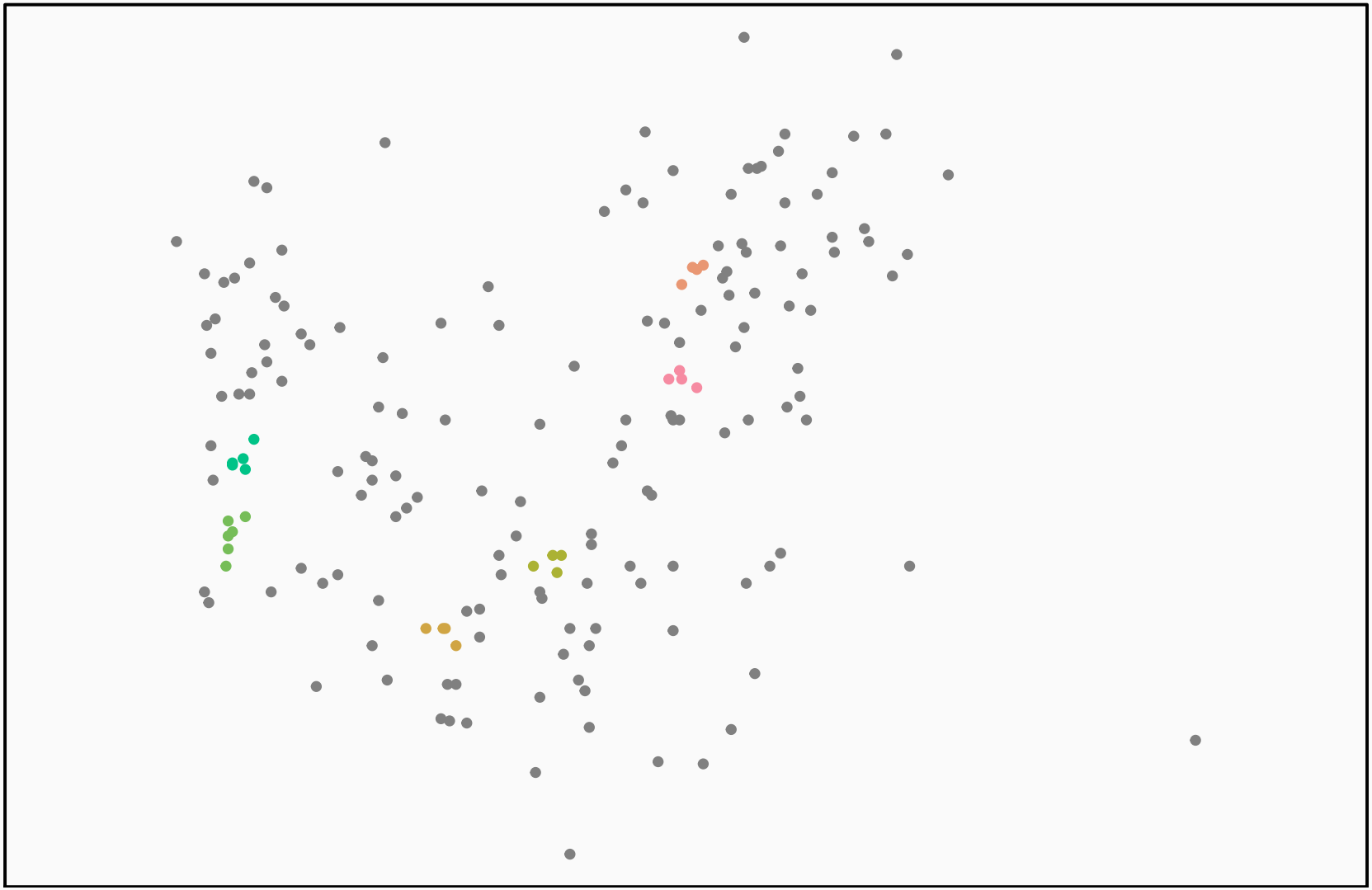
Clusters

$\varepsilon = 0.1325 \quad k = 11 \text{ noise} = 112$



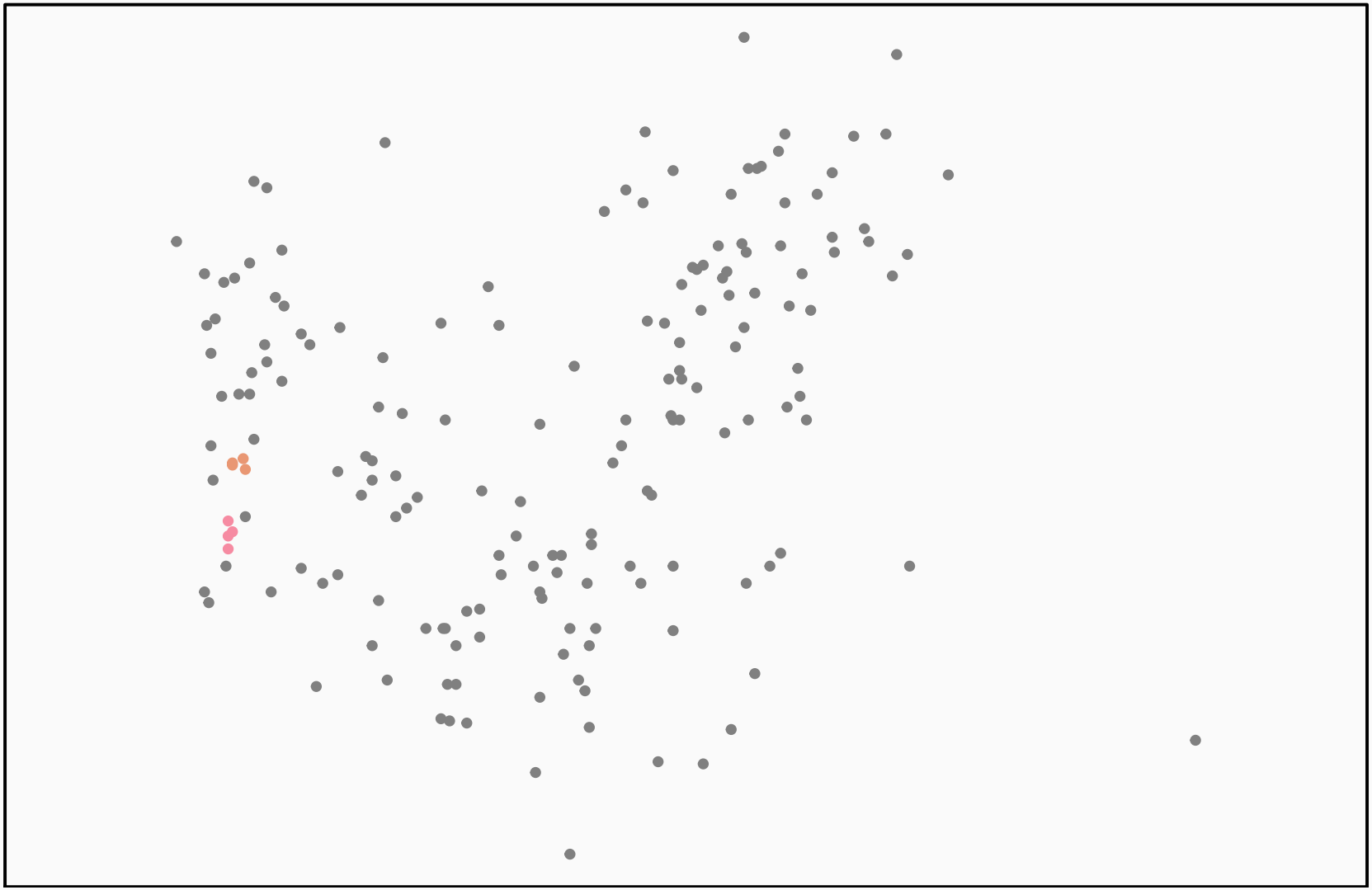
Clusters

$\varepsilon = 0.10375 \quad k = 6 \text{ noise} = 151$



Clusters

$\varepsilon = 0.075 \quad k = 2 \text{ noise} = 170$



Summary

DBSCAN

- ☺ noise detection
- ☺ arbitrary shaped clusters
- ☺ mostly deterministic
- ☹ $\Theta\left(N^2 P\right)$ running time in the worst case
- ☹ arbitrary shaped clusters
- ☹ very sensitive to the value of ε

Outline

Non crisp clusters

Ambiguous points

- some points are difficult to associate to a given cluster
- numerous situations: very close clusters, “interpolation” points, etc.

Soft clusters

- $\mathbf{X}_i$ belongs to C_k with “intensity”
- clustering through an assignment matrix: $(M_{ik})_{1 \leq i \leq N, 1 \leq k \leq K}$, with
  - $M_{ik} \in [0, 1]$: intensity between 0 and 1
  - $\sum_{k=1}^{K} M_{ik} = 1$: total unitary grade
  - crisp limit: $M_{ik} \in \{0, 1\}$
- interpretations:
  - M_{ik} as a membership grade: fuzzy sets
  - M_{ik} as a membership probability: mixture models

Fuzzy vector quantization

Vector quantization with membership

- prototypes: $\mathbf{\Gamma} = (\gamma_1, \ldots, \gamma_K)$
- assignment matrix: $(M_{ik})_{1 \leq i \leq N, 1 \leq k \leq K}$
- quality criterion

$$\mathcal{E}^b(\mathbf{\Gamma}, \mathbf{z}) = \sum_{i=1}^{N} \sum_{k=1}^{K} M_{ik}^b d(\mathbf{X}_i, \gamma_k),$$

Fuzziness parameter

- b represents the non crispness of the assignment
- $b = 1$ corresponds to the standard quantization problem
- $b > 1$ generates fuzzy assignments

Fuzzy c-means

Principle

- $\mathcal{E}^b(\mathbf{\Gamma}, \mathbf{z})$ is easily optimized with alternate optimization
- constrained optimization with respect to M

Algorithm

for $d(\mathbf{X}_i, \gamma_k) = \|\mathbf{X}_i - \gamma_k\|^2$

select $\mathbf{\Gamma}$ as a random subset of $\mathcal{D}$

repeat

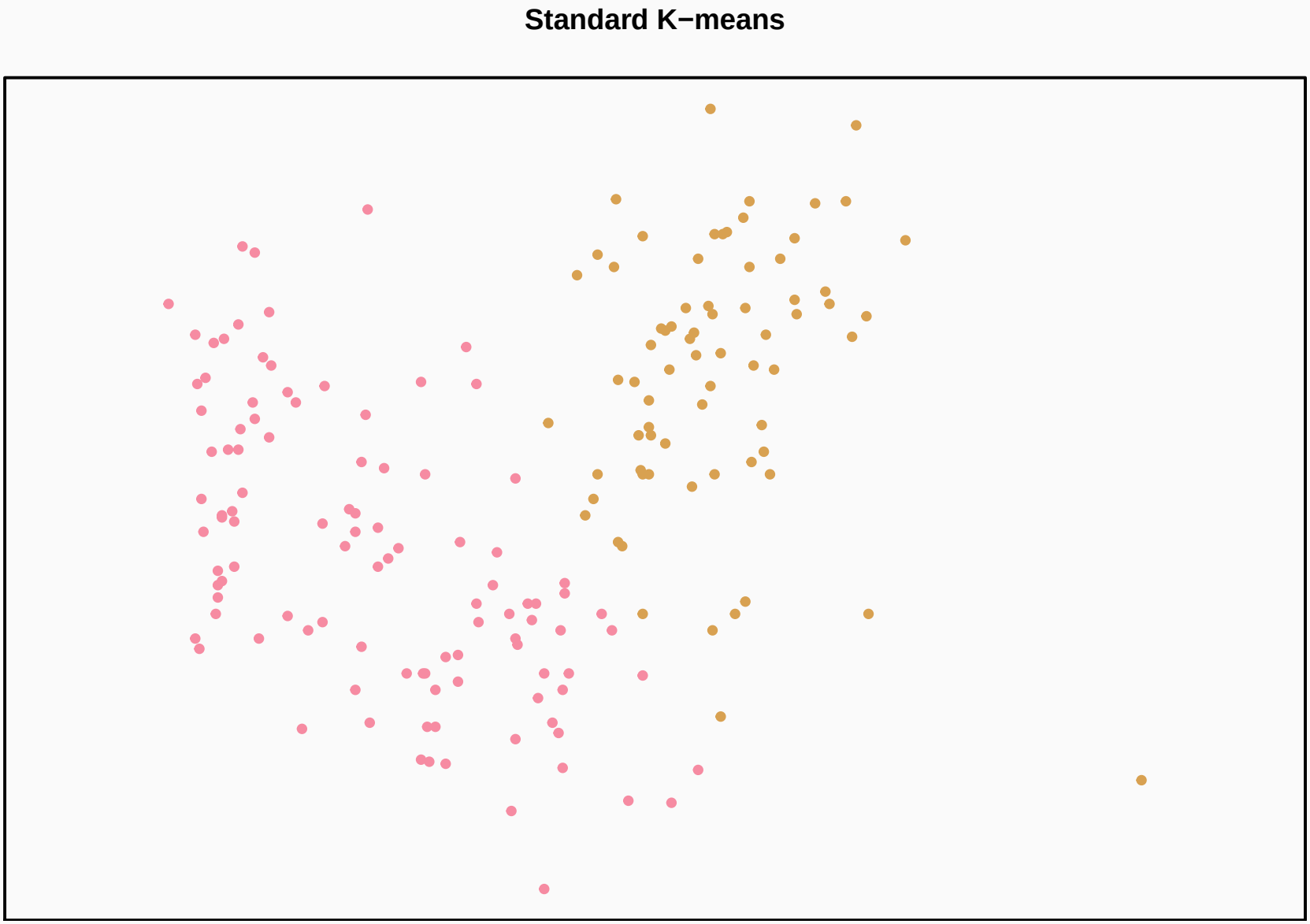
compute $d_{ik} = \|\mathbf{X}_i - \gamma_k\|^2$

$$M_{ik} = \frac{(1/d_{ik})^{1/(b-1)}}{\sum_{j=1}^{K}(1/d_{ij})^{1/(b-1)}} \qquad \triangleright \text{ assignment phase}$$

$$\gamma_k \leftarrow \frac{1}{\sum_{j=1}^{N} M_{jk}^b} \sum_{i=1}^{N} M_{ik}^b \mathbf{X}_i \qquad \triangleright \text{ representation phase}$$

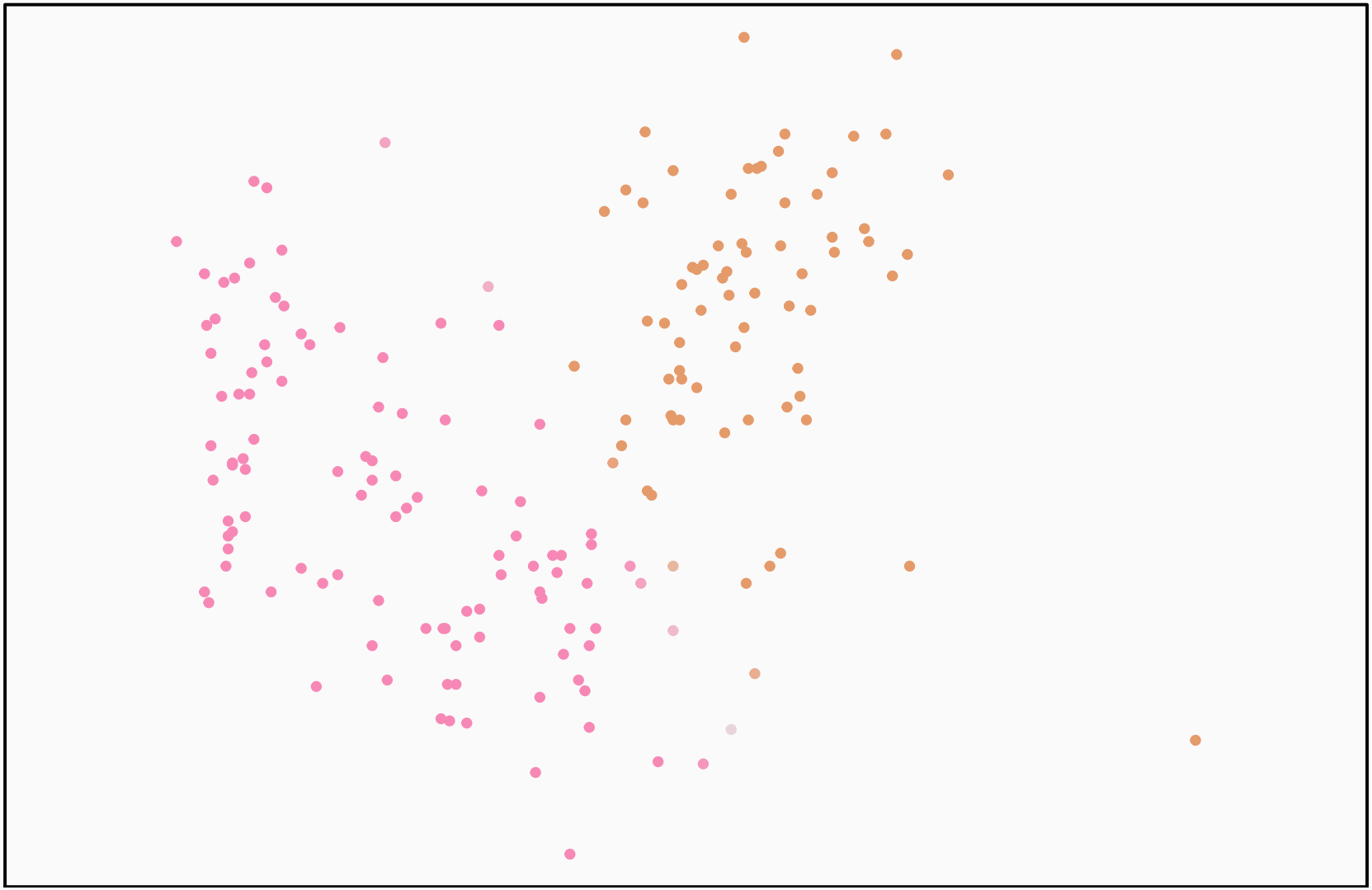
until convergence

Examples



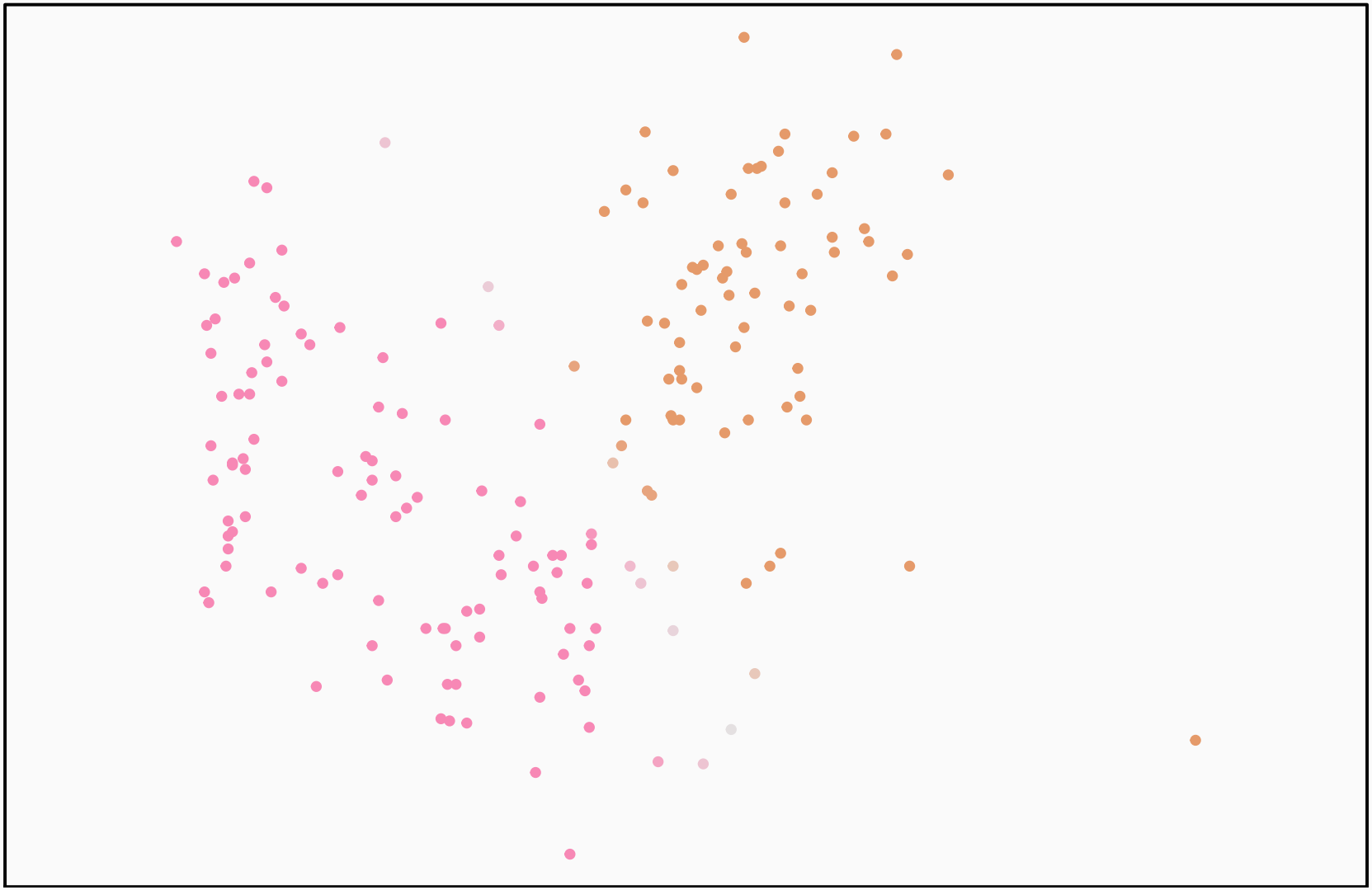
Examples

b = 1.1



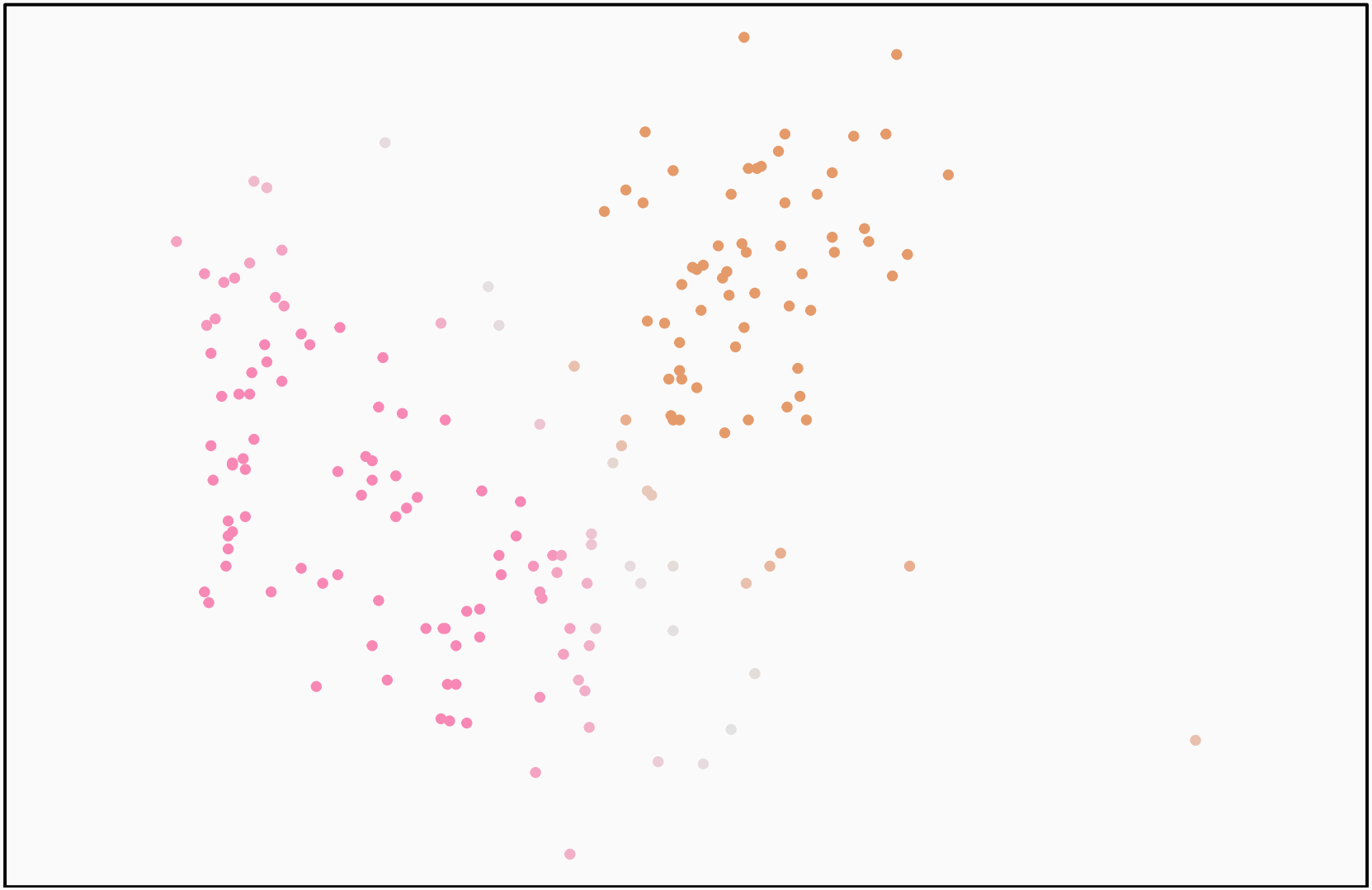
Examples

$b = 1.2$



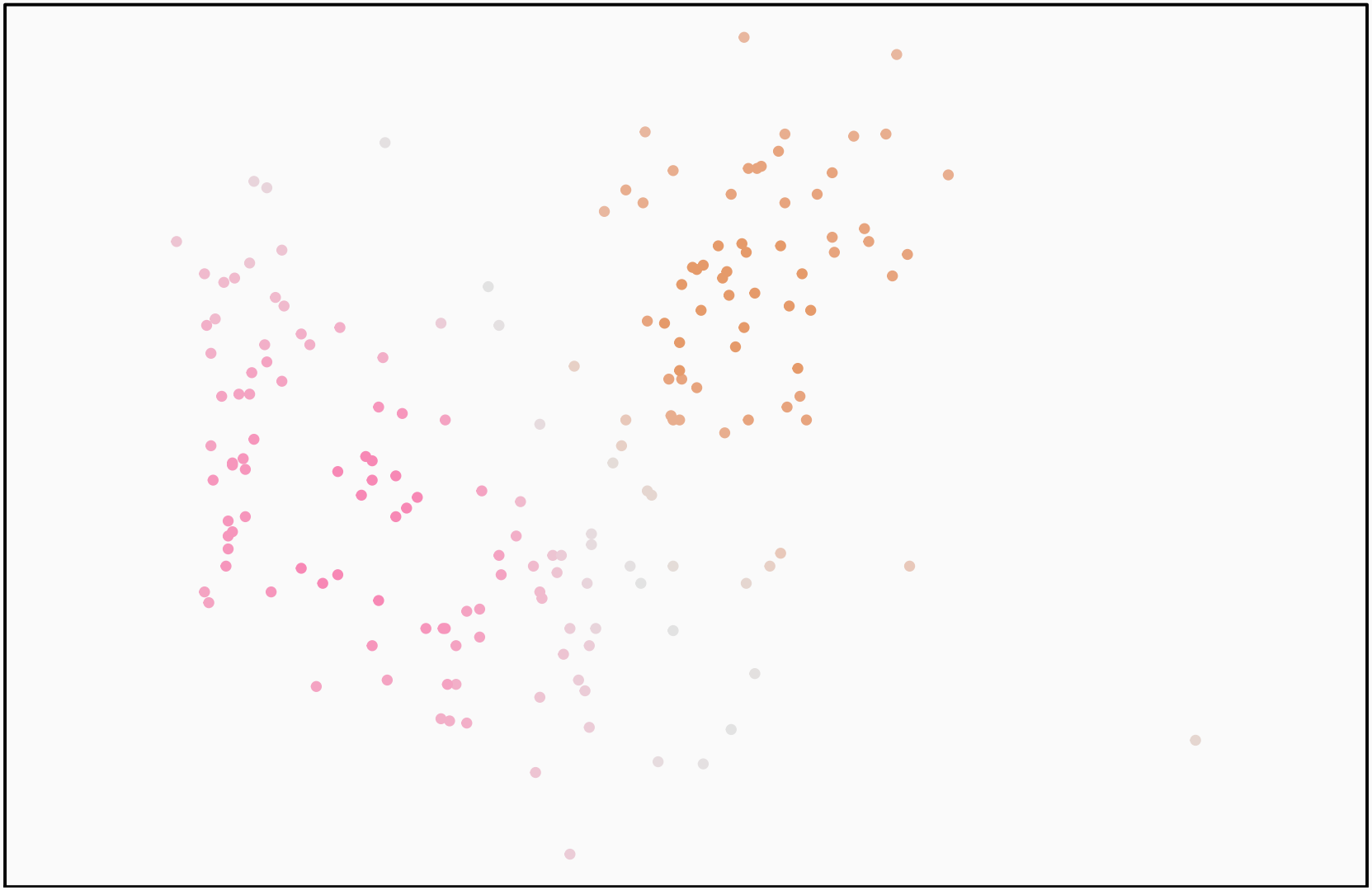
Examples

$b = 1.5$



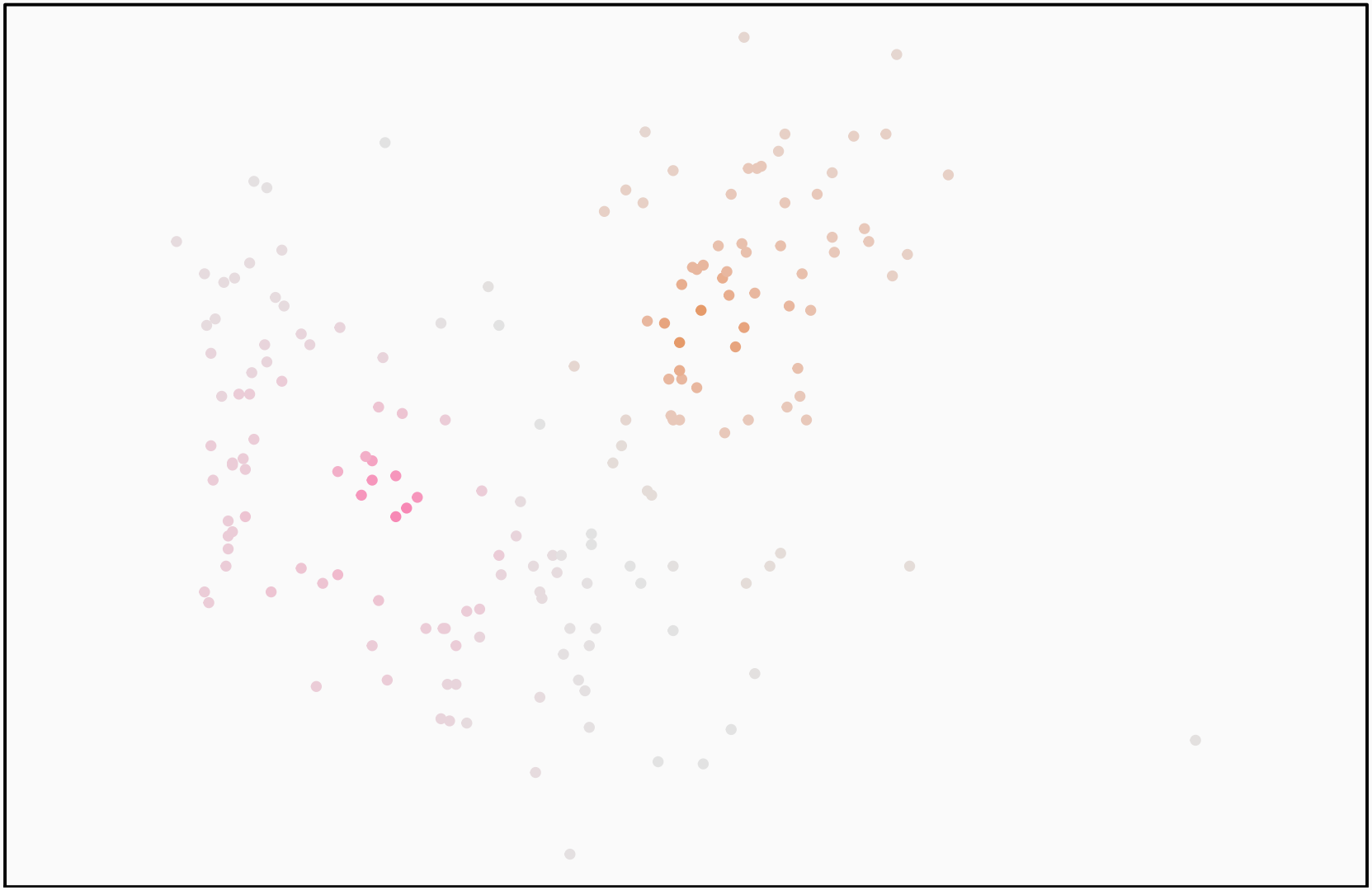
Examples

$b = 2$



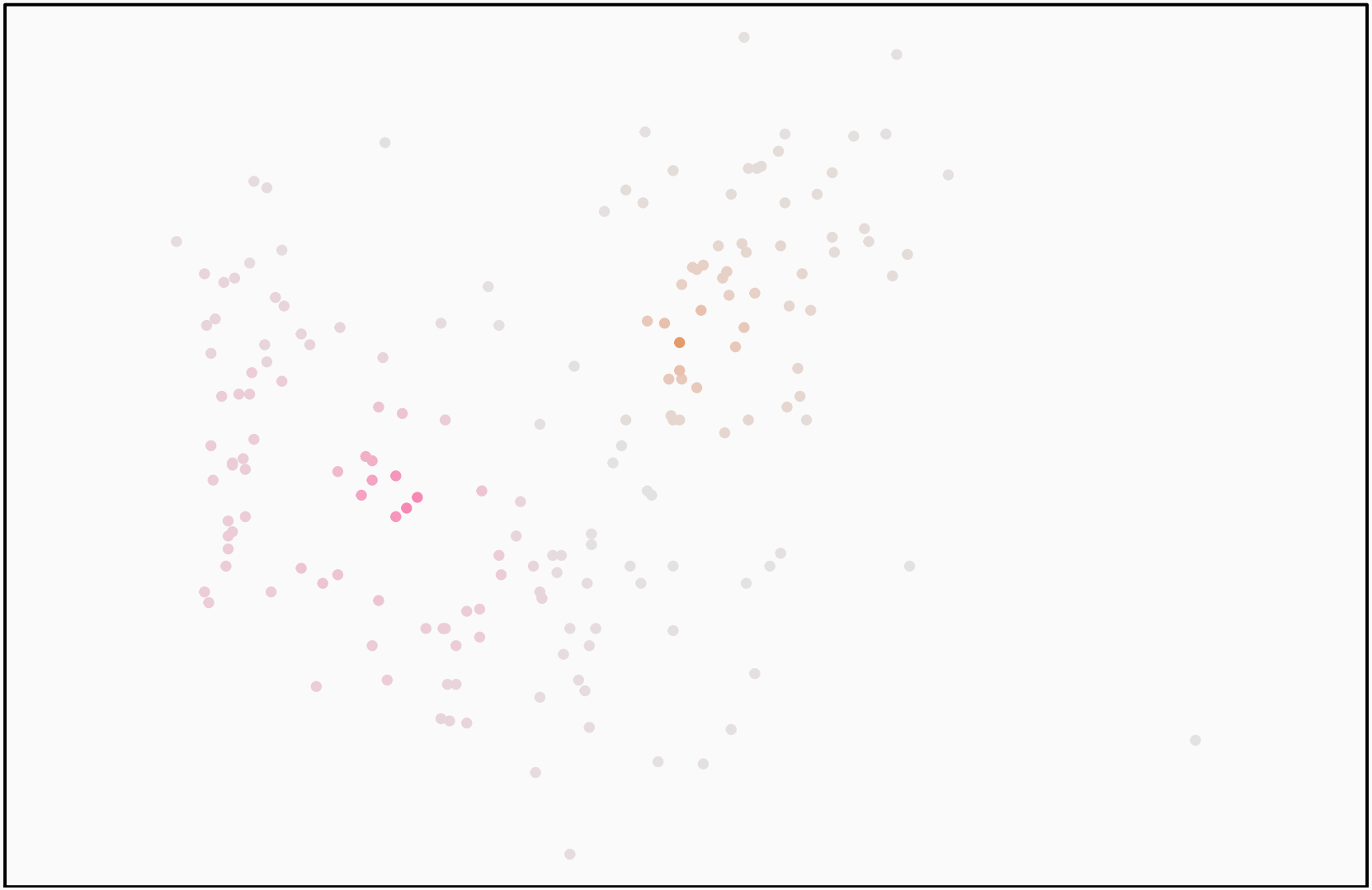
Examples

$b = 5$

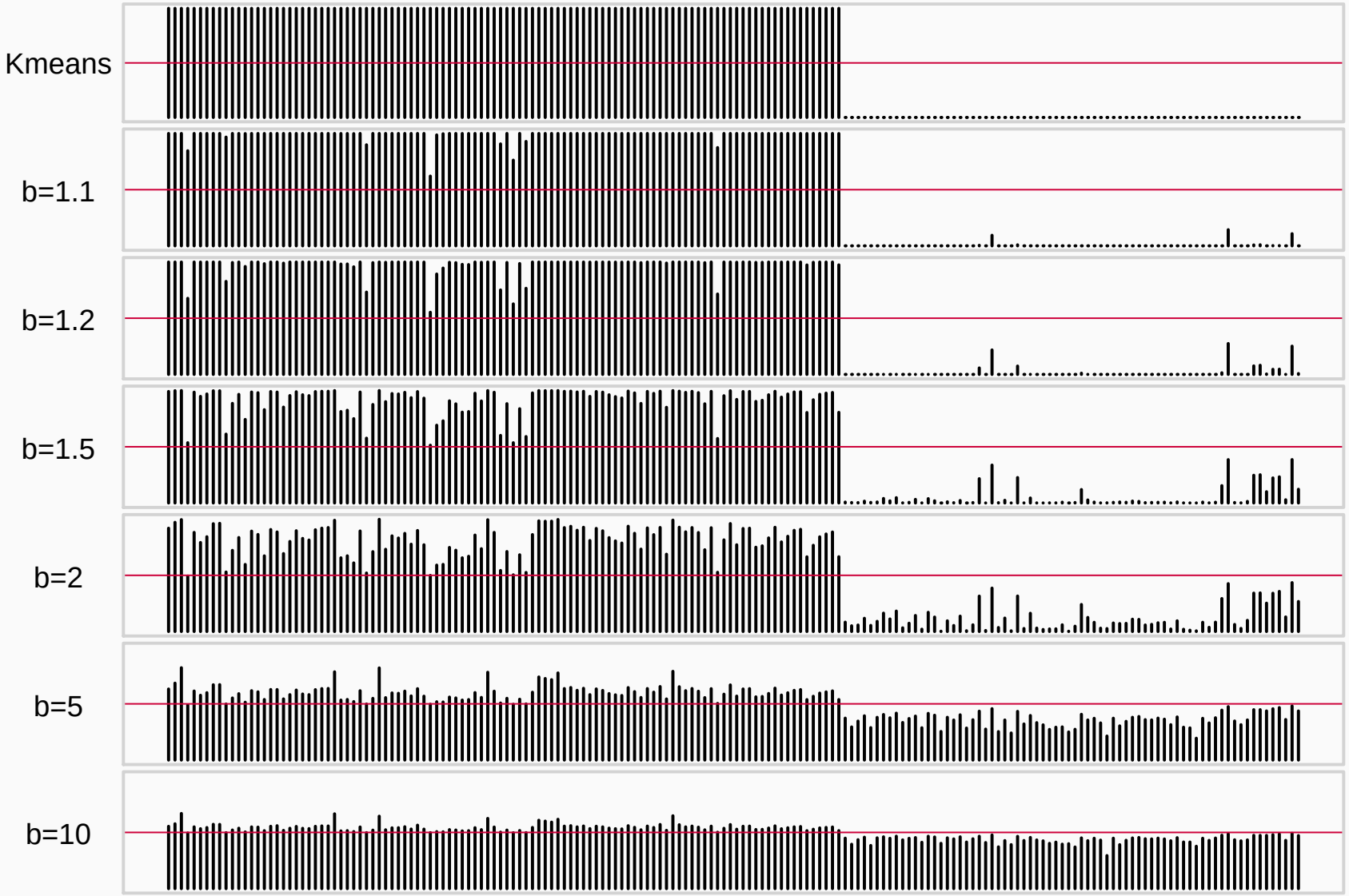


Examples

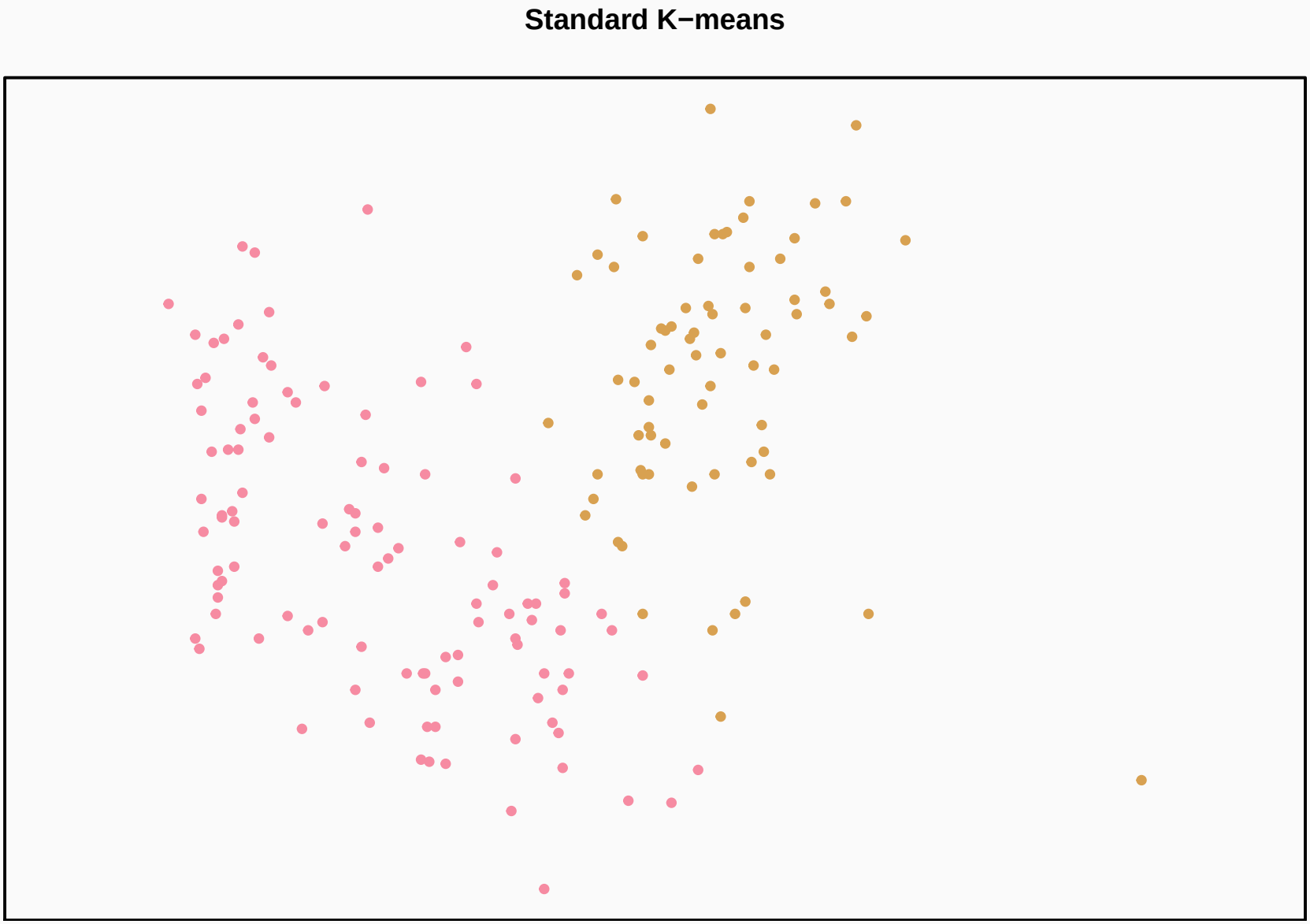
$b = 10$



Fuzziness evolution

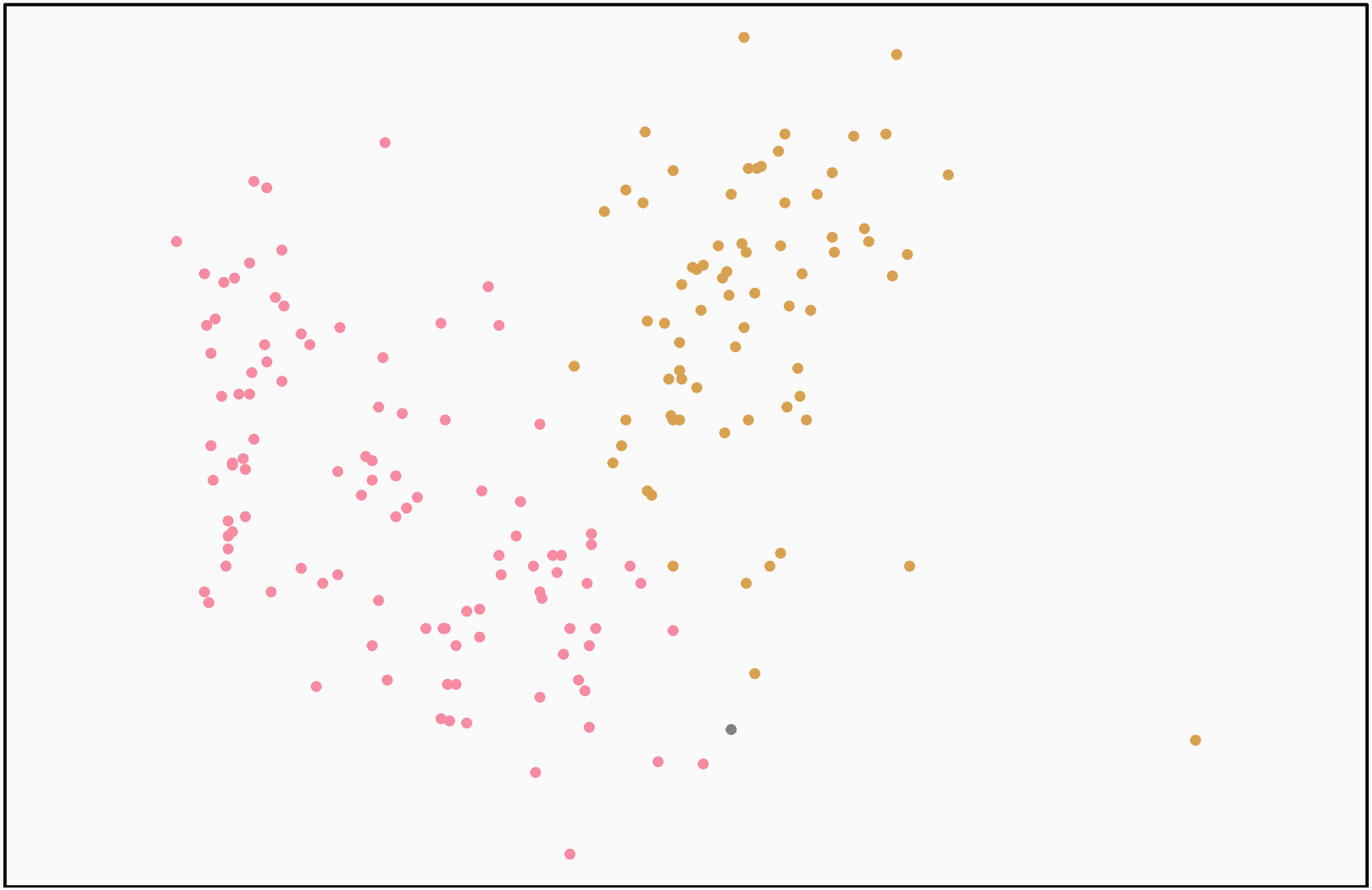


Defuzzified clustering



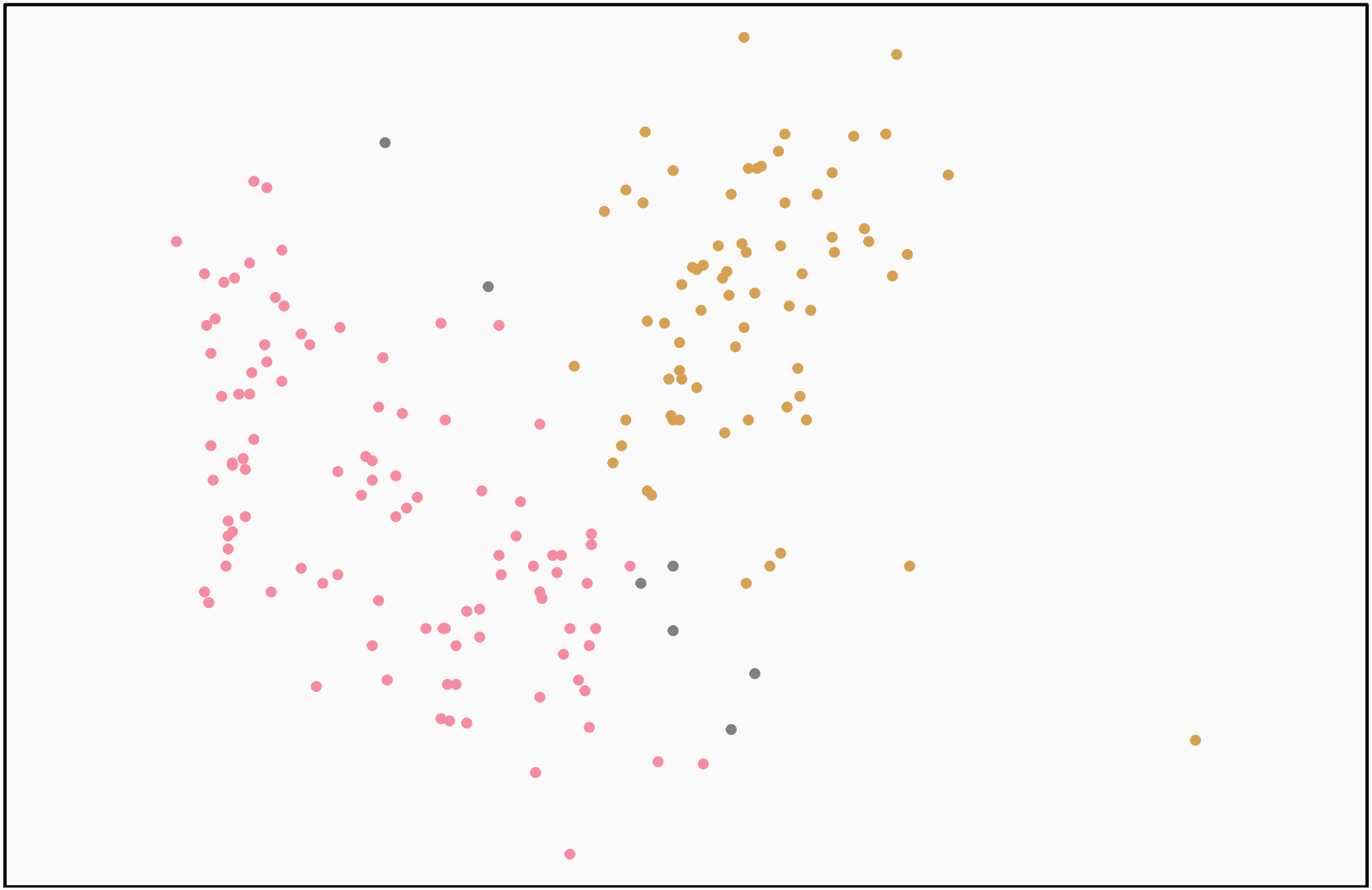
Defuzzified clustering

b = 1.1



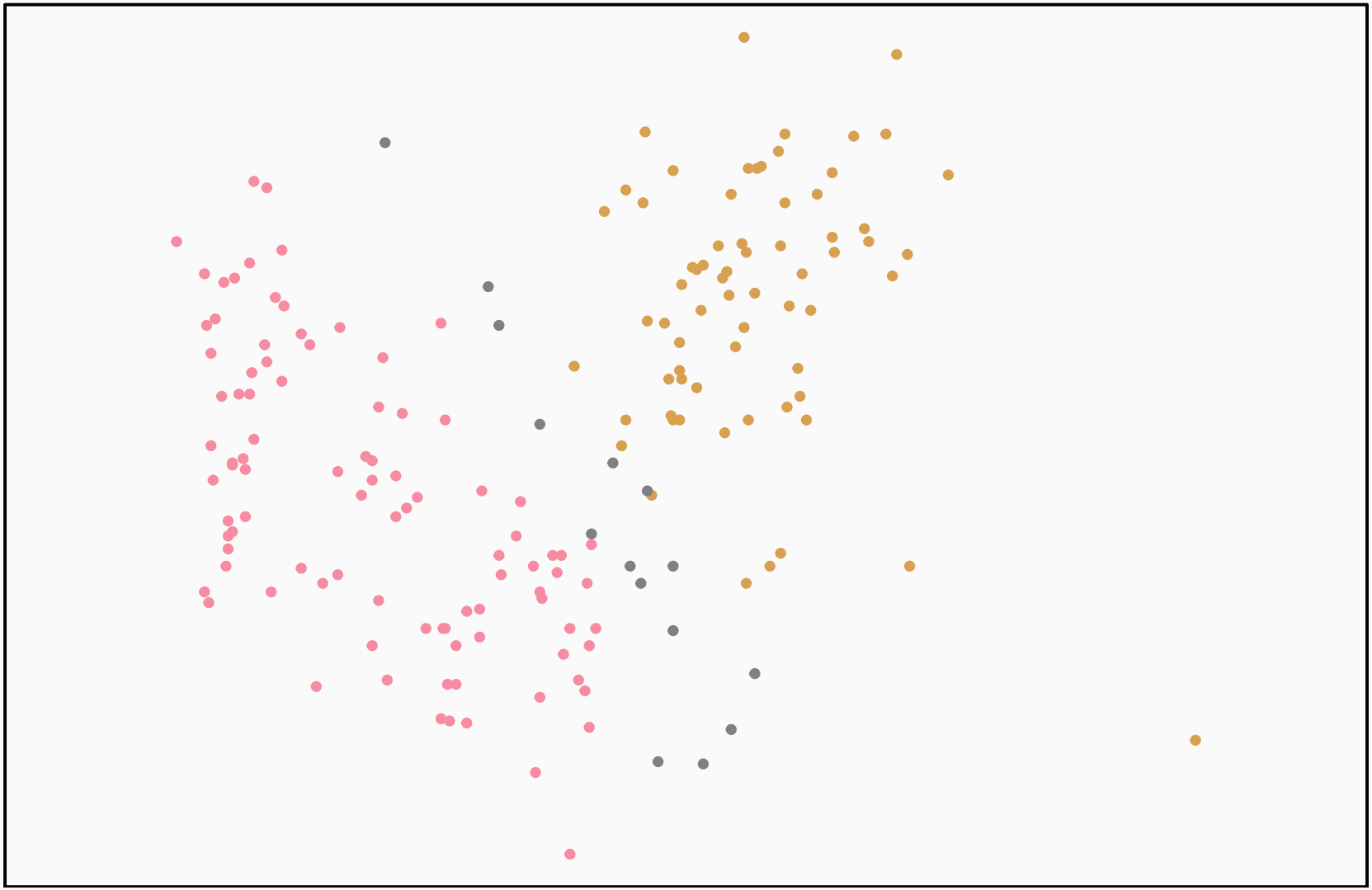
Defuzzified clustering

b = 1.2



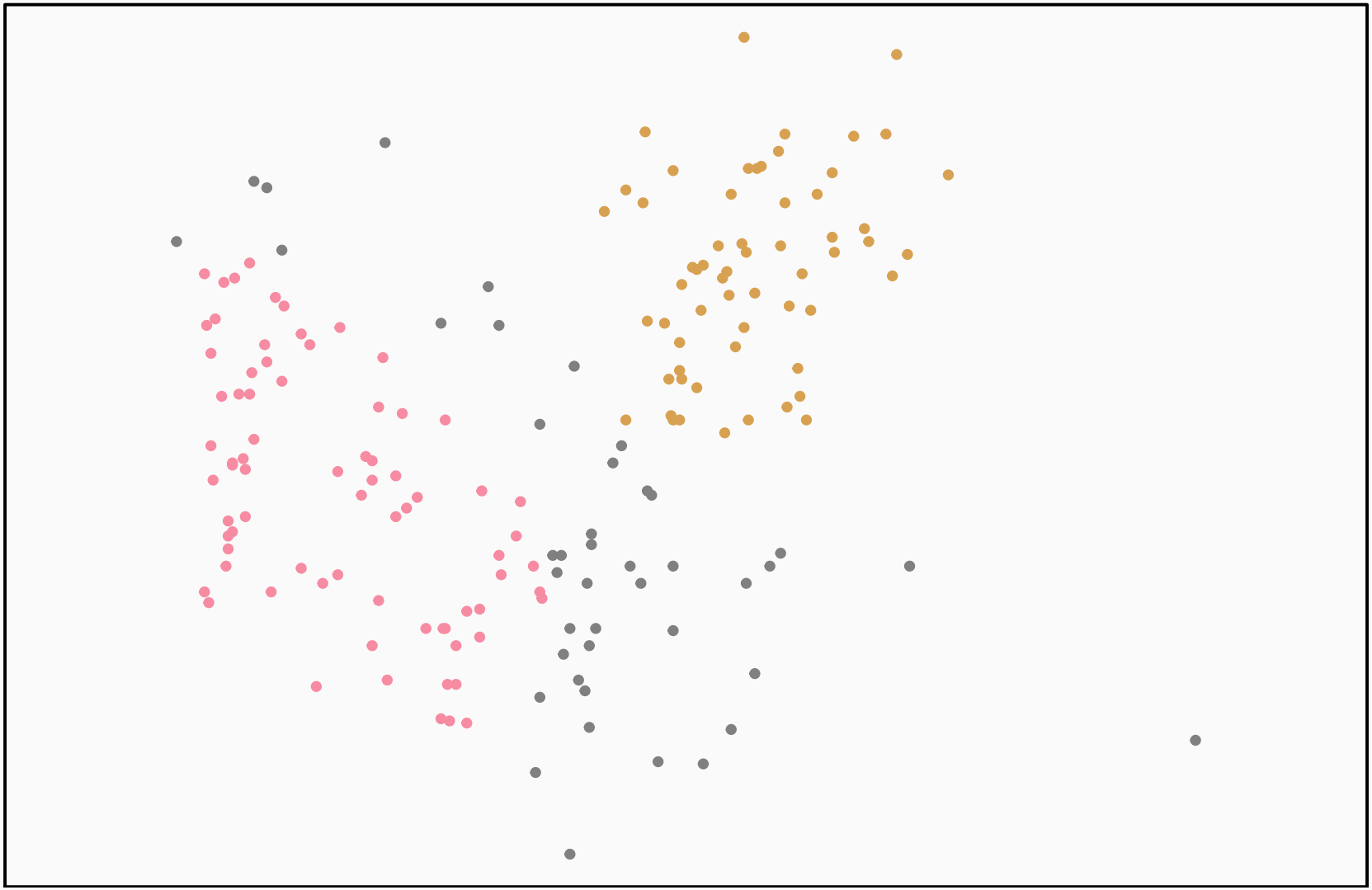
Defuzzified clustering

b = 1.5



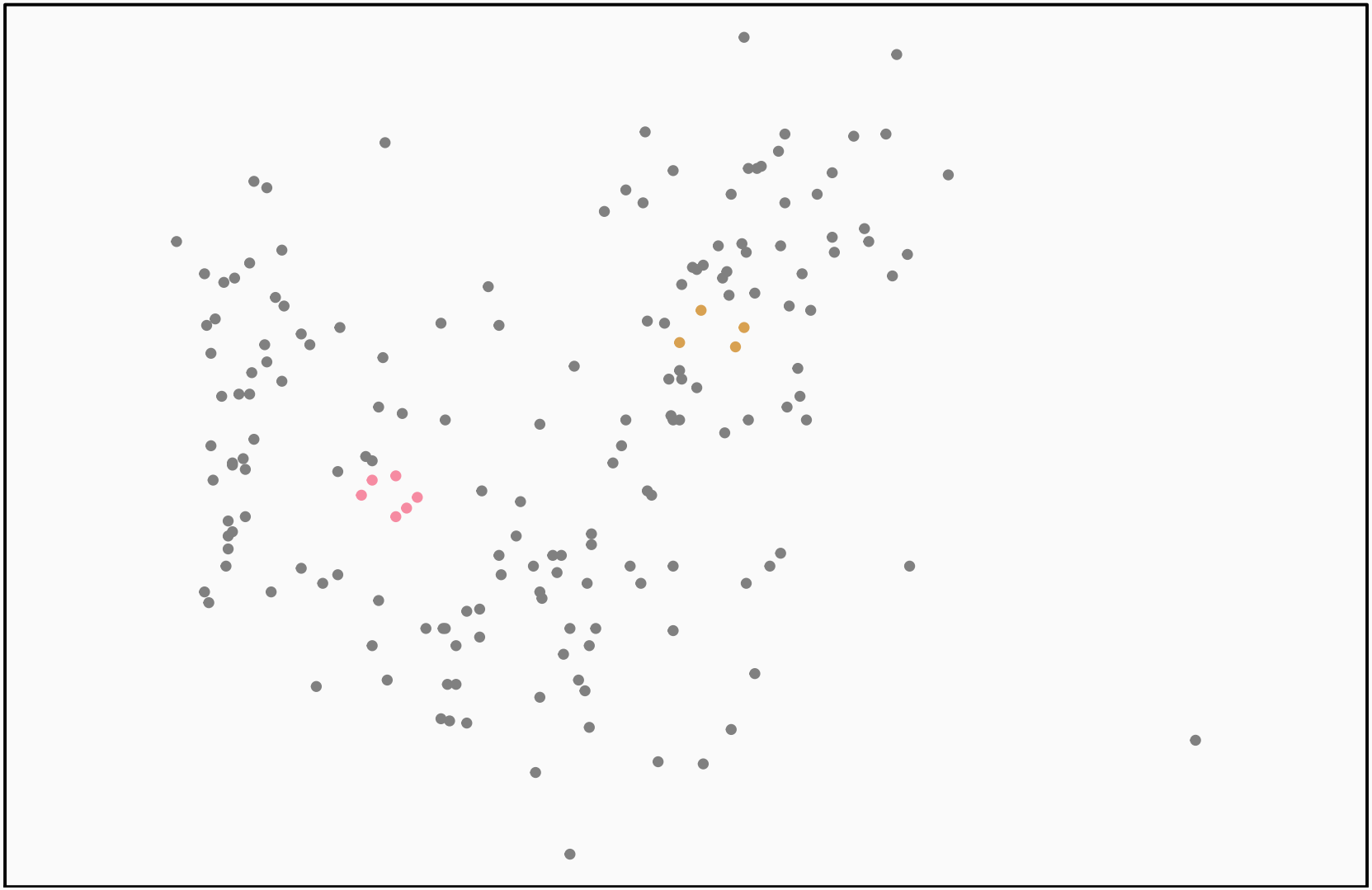
Defuzzified clustering

b = 2



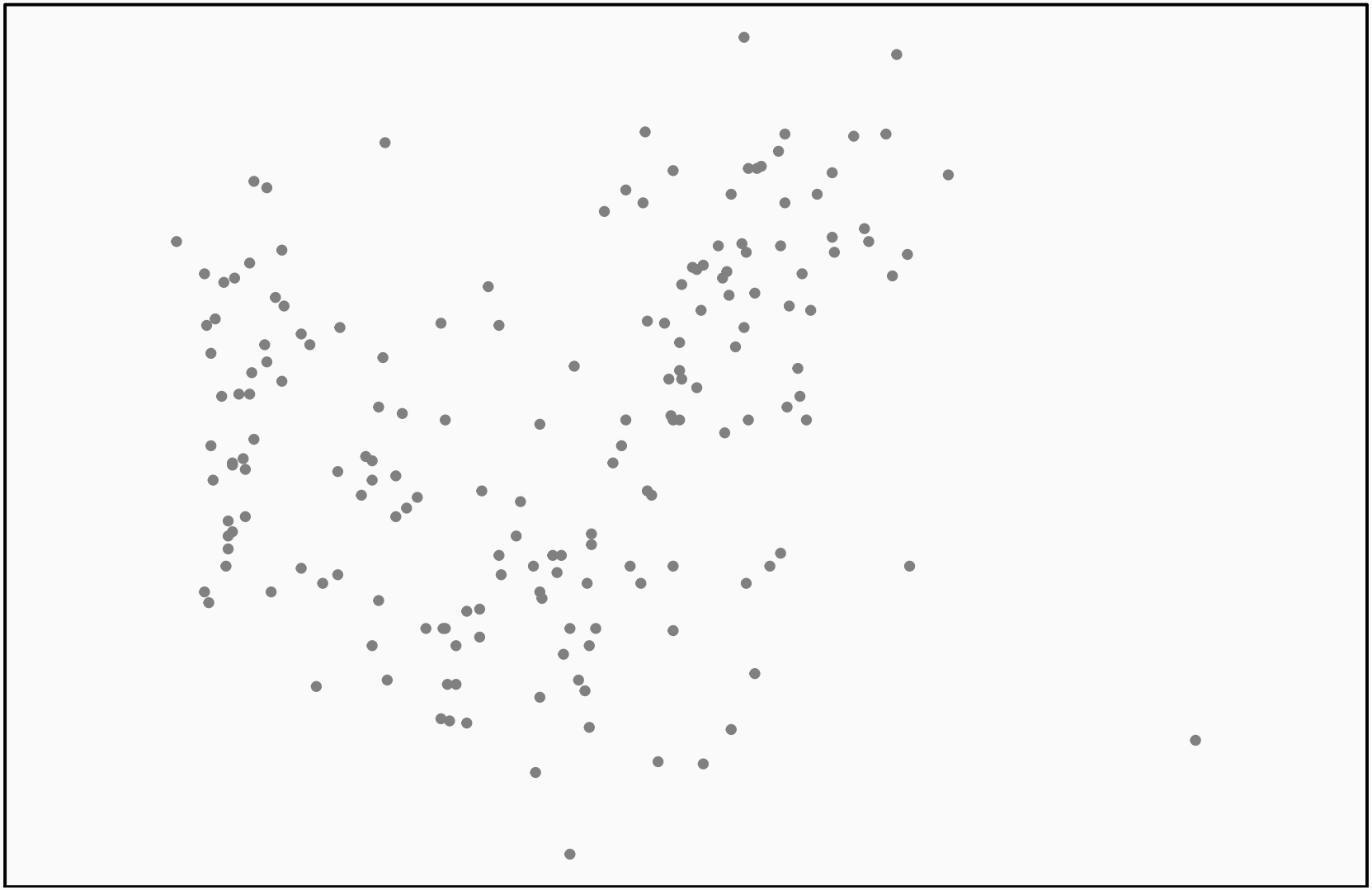
Defuzzified clustering

$b = 5$



Defuzzified clustering

$b = 10$



Summary

Fuzzy c-cmeans

- ☺ soft clustering
- ☺ simple interpretation and implementation
- ☺ identify points that are complex to cluster
- ☹ slightly more expansive than the k-means
- ☹ quite sensitive to the additional parameter b

Strategy

- test several values of b
- use diagnostic plots
- can be used to identify core points

Fuzzy versus probabilistic

M_{ik} as a membership grade

- intrinsic fuzziness
- clusters are inherently ill defined
- no randomness

M_{ik} as a probability

- missing information
- belief
- clusters are perfectly defined but unknown to the analyst

Probabilistic approach

Generative models

- parametric model for the distribution of $(\mathbf{X}_i)_{1 \leq i \leq N}$
- parameter estimation from a data set via maximum likelihood (or other techniques)

Clustering oriented models

- mixture models
- K parametric models with prior probabilities π_k ($\sum_{k=1}^{K} \pi_k = 1$)
- generative process for $\mathbf{X}_i$
  1. chose $z_i \in \{1, \ldots, K\}$ with probability π_k for k
  2. generate $\mathbf{X}_i$ according to the parametric model z_i

Formal model

Hidden variable model

- each observation $\mathbf{X}_i$ is associated to a hidden variable Z_i
- each Z_i takes values in $\{1, \dots, K\}$, with $\mathbb{P}(Z_i = k) = \pi_k$
- the Z_i are independent and given $Z_1, \dots, Z_N$, the $\mathbf{X}_i$ are independent
- we are given K parametric distributions $(p_k)_{1 \leq k \leq K}$ on $\mathcal{X}$ with parameters $\theta = (\theta_k)_{1 \leq k \leq K}$
- $\mathbf{X}_i \mid Z_i = k$ is distributed according to p_k
- then the log likelihood of the data set $\mathcal{D} = (\mathbf{x}_i)_{1 \leq i \leq N}$ is given by

$$\log p(\mathcal{D} \mid \pi, \theta) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k p_k(\mathbf{x}_i \mid \theta_k) \right)$$

Complete likelihood

Notations

- integer notation $Z_i \in \{1, \dots, K\}$
- binary notation $Z_i \in \{0, 1\}^K$ with $\sum_{k=1}^K Z_{ik} = 1$
- $Z_i = k \Leftrightarrow Z_{ik} = \delta_{ik}$
- then $p(z_i \mid \pi) = \prod_{k=1}^K \pi_k^{z_{ik}}$

Complete likelihood

- the log likelihood of the full data set $\mathcal{D}_F = (\mathbf{x}_i, z_i)_{1 \leq i \leq N}$ is given by

$$\log p(\mathcal{D}_F \mid \pi, \theta) = \sum_{i=1}^N \sum_{k=1}^K z_{ik} \left(\log \pi_k + \log p_k(\mathbf{x}_i \mid \theta_k)\right)$$

- easy to optimize compare to $\log p(\mathcal{D} \mid \pi, \theta)$

Guessing the hidden variables

Reversing the model

- according to the Bayes rule

$$\mathbb{P}(Z_i = k \mid \mathbf{x}_i, \pi, \theta) = \frac{p_k(\mathbf{x}_i \mid \theta_k)\pi_k}{\sum_{l=1}^{K} \pi_l p_l(\mathbf{x}_i \mid \theta_l)}$$

- $\gamma_{ik} = \mathbb{P}(Z_i = k \mid \mathbf{x}_i, \pi, \theta)$ is the responsibility of component k for generating $\mathbf{x}_i$
- z_i can be "guessed" in a probabilistic sense given the true parameters of the model

EM algorithm

Key ideas

- averaging $\sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \left(\log \pi_k + \log p_k(\mathbf{x}_i \mid \theta_k) \right)$ over the probabilistic guesses of the z_i
- using the best possible guess $\gamma_{ik} = \mathbb{P}(Z_i = k \mid \pi, \theta)$
- alternating between improving the estimates of the parameters π and θ and improving the estimates of the hidden variables γ_{ik}

EM algorithm

initialize $\pi^{(0)}$ and $\theta^{(0)}$
$t \leftarrow 1$
repeat
compute ▷ E Phase

$$\gamma_{ik}^{(t)} = \frac{p_k(\mathbf{x}_i \mid \theta_k^{(t-1)})\pi_k^{(t-1)}}{\sum_{l=1}^{K} \pi_l^{(t-1)} p_l(\mathbf{x}_i \mid \theta_l^{(t-1)})}$$

compute ▷ M Phase

$$N_k^{(t)} = \sum_{i=1}^{N} \gamma_{ik}^{(t)} \qquad \pi_k^{(t)} = \frac{N_k^{(t)}}{N}$$

$$\theta_k^{(t)} = \arg\max_{\theta_k} \sum_{i=1}^{N} \gamma_{ik}^{(t)} \log p_k(\mathbf{x}_i \mid \theta_k)$$

$t \leftarrow t + 1$
until convergence

Gaussian mixture

Standard approach for $\mathcal{X} = \mathbb{R}^P$

- each p_k is a multivariate Gaussian distribution
- $\theta_k = (\mu_k, \Sigma_k)$ and

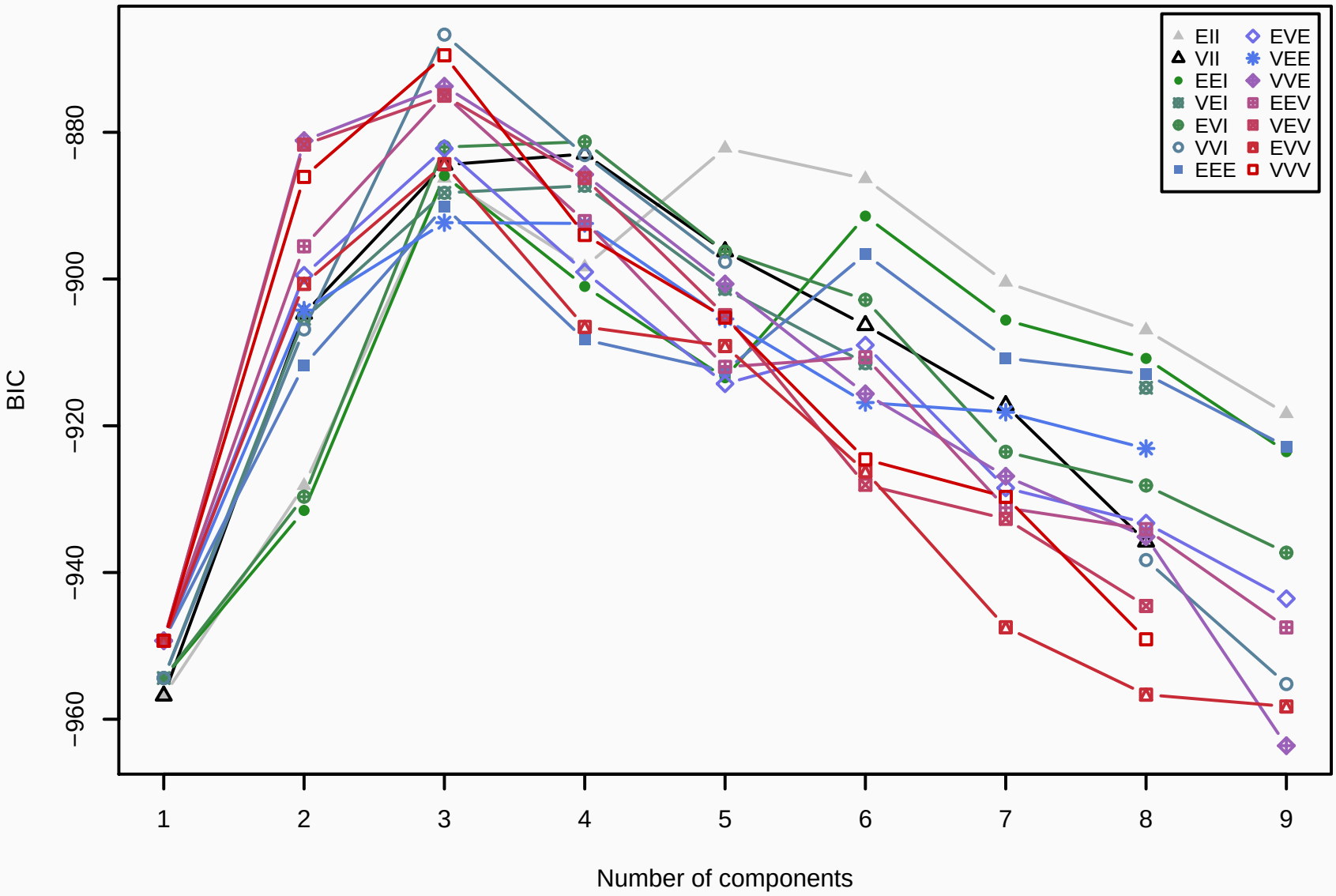
$$p(\mathbf{x}_i|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{P/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}_i-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}_i-\mu_k)}$$

- then we have

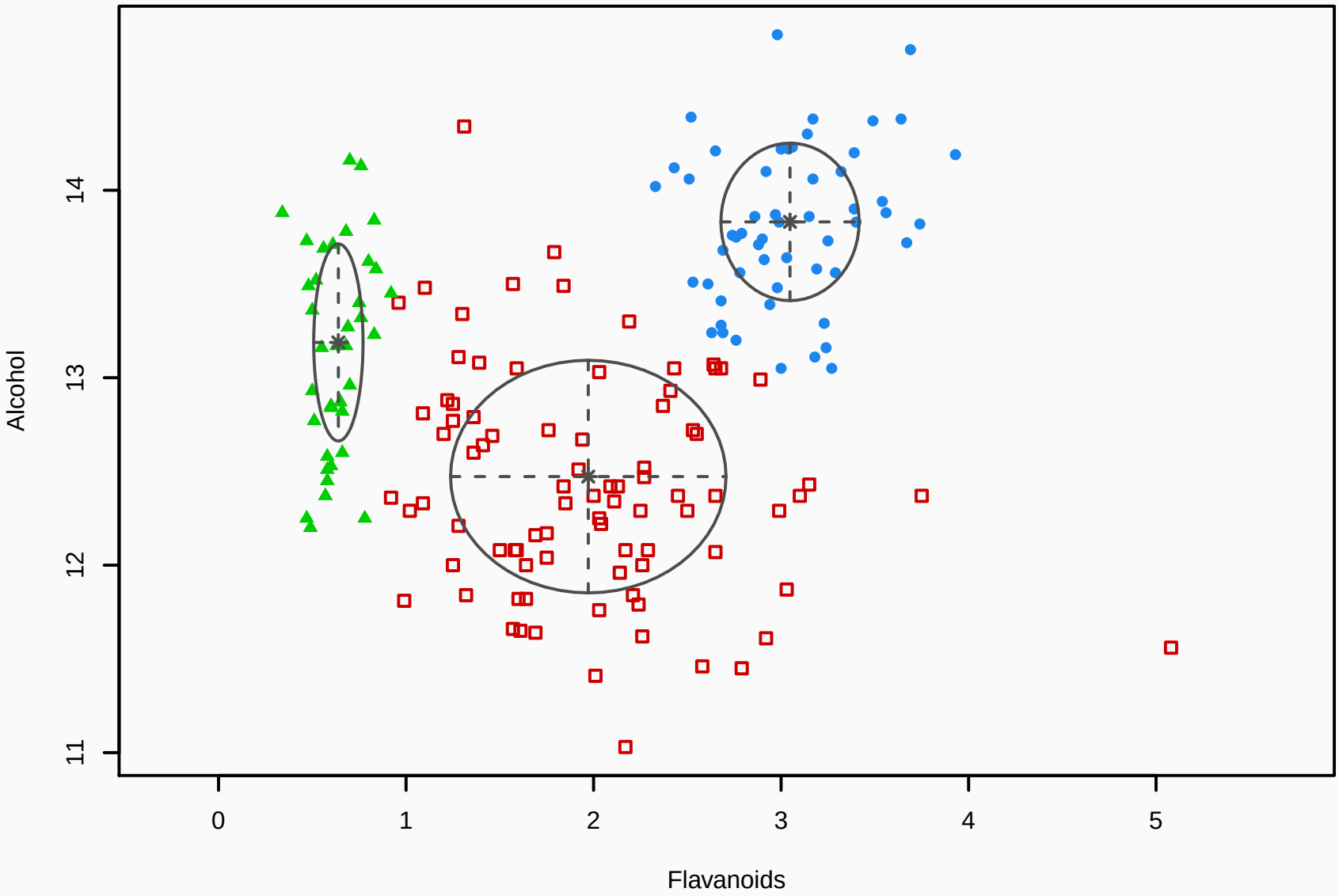
$$\mu_k^{(t)} = \frac{1}{N_k^{(t)}} \sum_{i=1}^{N} \gamma_{ik}^{(t)} \mathbf{x}_i$$

$$\Sigma_k^{(t)} = \frac{1}{N_k^{(t)}} \sum_{i=1}^{N} \gamma_{ik}^{(t)} (\mathbf{x}_i - \mu_k^{(t)})^T (\mathbf{x}_i - \mu_k^{(t)})$$

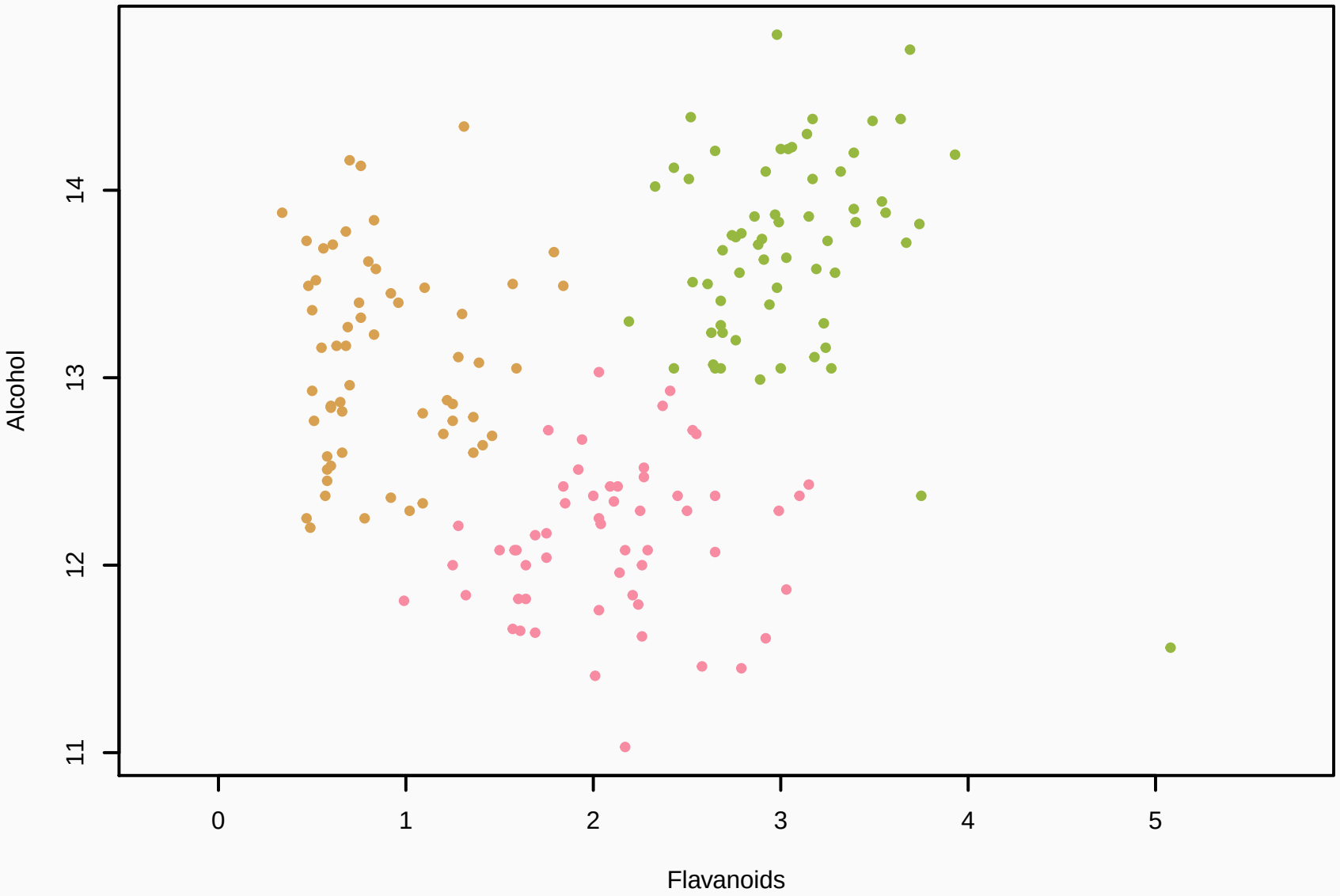
Example


EII
VII
EEI
VEI
EVI
VVI
EEE
EVE
VEE
VVE
EEV
VEV
EVV
VVV
BIC
−880
−900
−920
−940
−960
1
2
3
4
5
6
7
8
9
Number of components

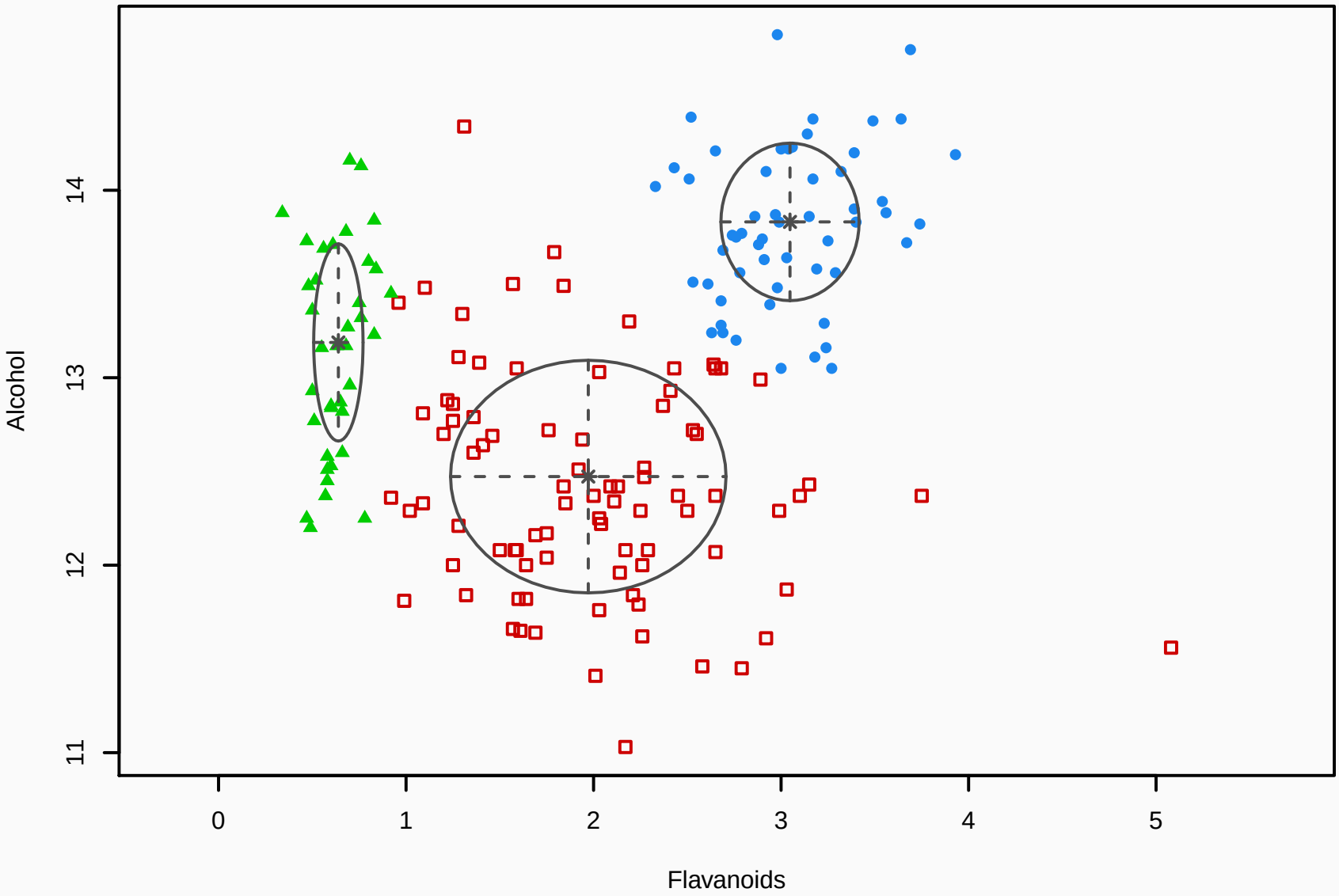
Example



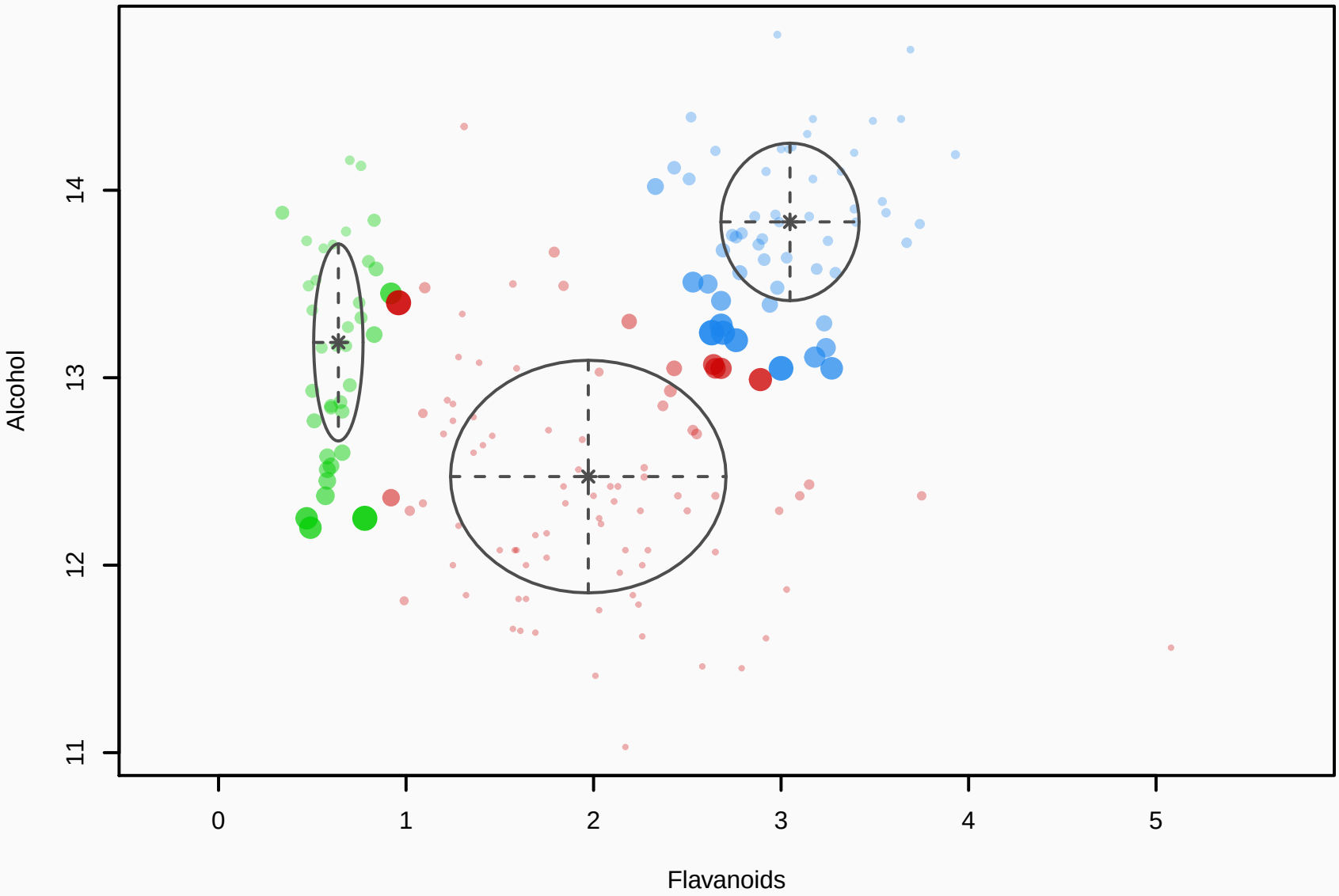
Example



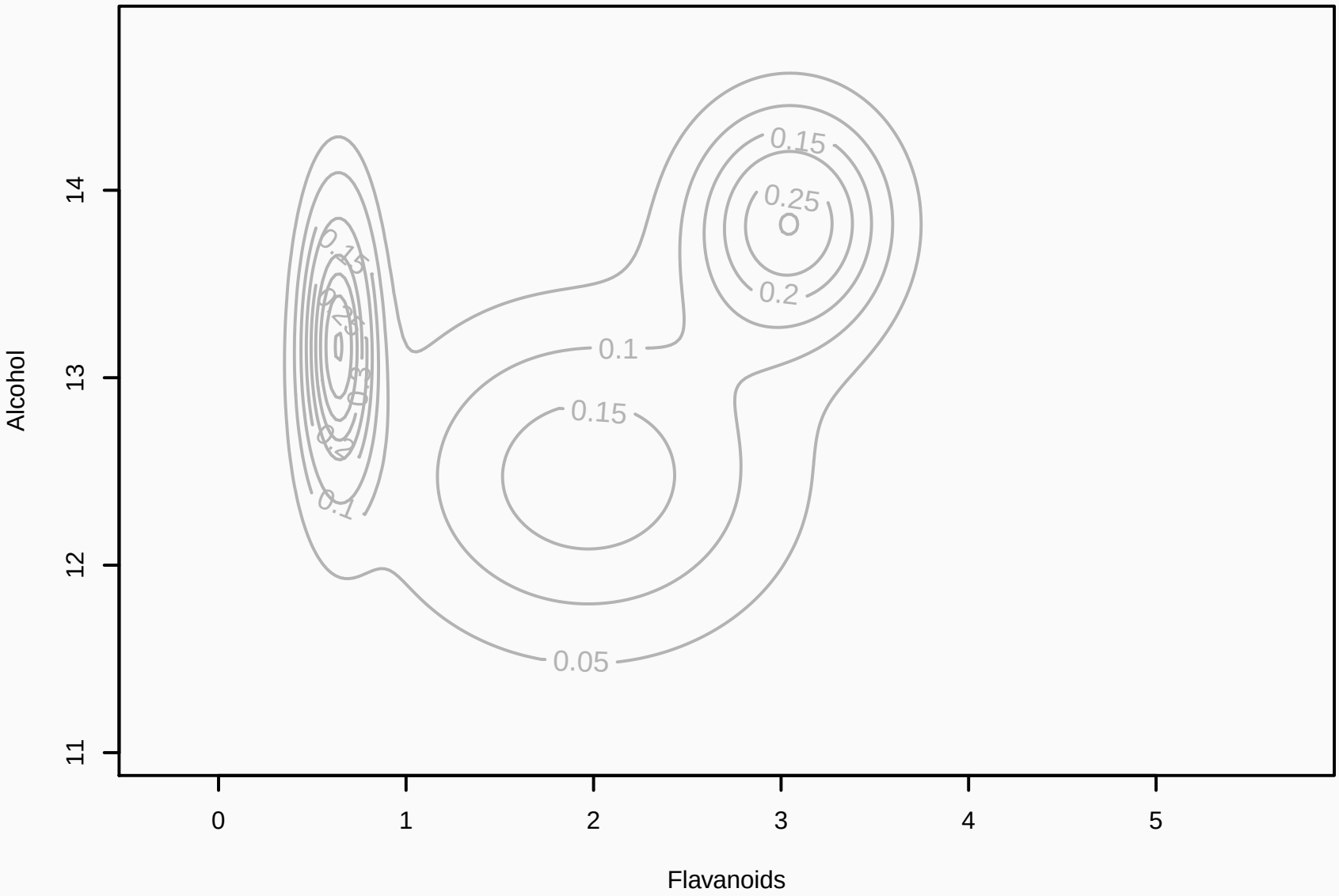
Example



Example



Example


Alcohol
14
13
12
11
0
1
2
3
4
5
Flavanoids
0.15
0.25
0.2
0.1
0.15
0.05
0.15
0.25
0.2
0.1

Summary

Mixture models

- ☺ soft clustering
- ☺ rich outputs
- ☺ automatic model selection (via BIC)
- ☺ very flexible framework
- ☹ somewhat complex implementation
- ☹ high computational cost in some cases

Conclusion

References I

D. Arthur and S. Vassilvitskii.

K-means++: The advantages of careful seeding.
In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu.

A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise.
In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

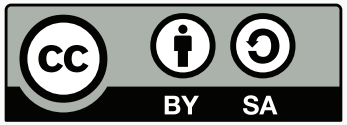
J. M. Kleinberg.

An impossibility theorem for clustering.
In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 463–470. MIT Press, 2003.

U. von Luxburg, R. C. Williamson, and I. Guyon.

Clustering: Science or art?
In I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 65–79, Bellevue, Washington, USA, 02 Jul 2012. PMLR.

Licence



Version

Last git commit: 2021-01-19
By: Fabrice Rossi (Fabrice.Rossi@apiacoa.org)
Git hash: 97cfd0a9975cf193f5790845c00e476c1572a327

Changelog

- April 2019:
  - added k-means++
  - added DBSCAN
  - added fuzzy c-means
  - added mixture models
- March 2018: initial version