Discussion Week 5

Reminders

- Deadlines
  - HW2 out: Due Nov 10 (Friday) 11:59 PM
  - Midterm report: Due Nov 5 (Sunday) 11:59 PM
- No discussion next week
  - Veterans Day Holiday (Nov 10)
- Midterm coming up!
  - Nov 13 (Monday)

Overview

- Clustering
  - Review
  - Applications
- Density Estimation

Clustering

- Goal: Automatically segment data into groups of similar points
- Question: When and why would we want to do this?
- Useful for:
  - Automatically organizing data
  - Understanding hidden structure in some data
  - Representing high-dimensional data in a low-dimensional space
- Examples:
  - Customers according to purchase histories
  - Genes according to expression profile
  - Search results according to topic
  - MySpace users according to interests
  - A museum catalog according to image similarity

Clustering

- Our data

$$D = \{x_1, \ldots, x_N\}$$

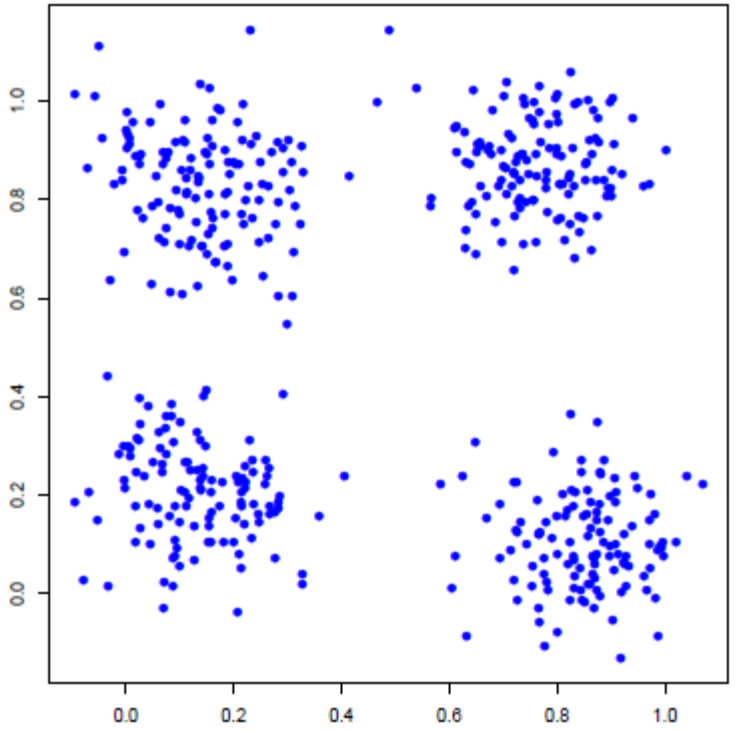
- Each data point is p-dimensional, i.e.,

$$x_n = < x_{n1}, \ldots, x_{np} >$$

- Define a distance function between data, $d(x_n, x_m)$.
- Goal: segment the data into k groups

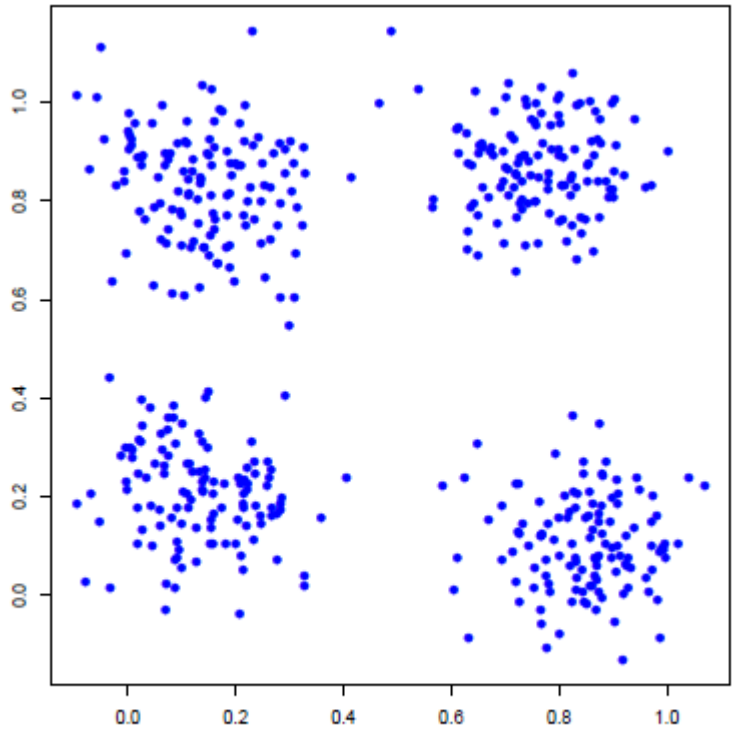
$$\{z_1, \ldots, z_N\} \text{ where } z_i \in \{1, \ldots, K\}.$$

Example data



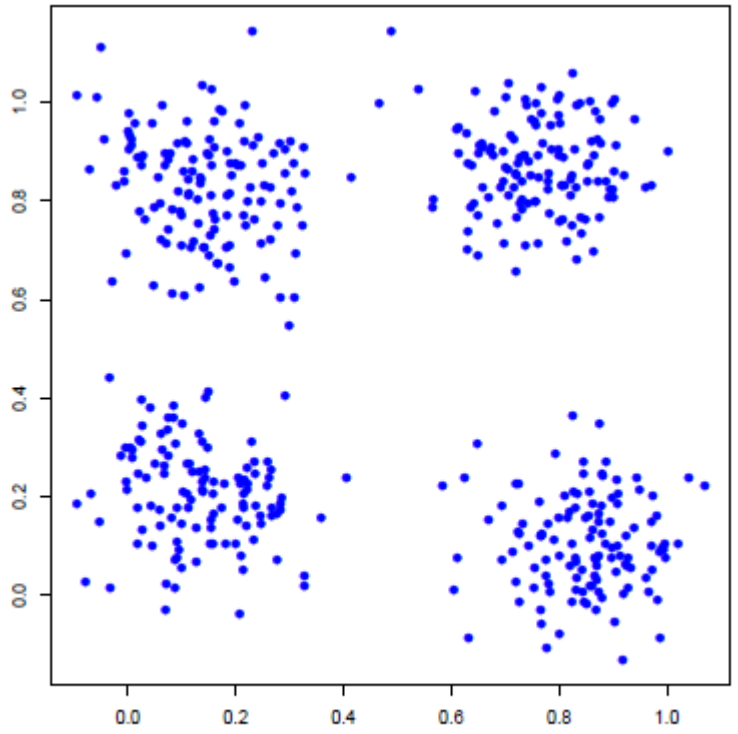
500 2-dimensional data points: $x_n = < x_{n1}, x_{n2} >$

Example data



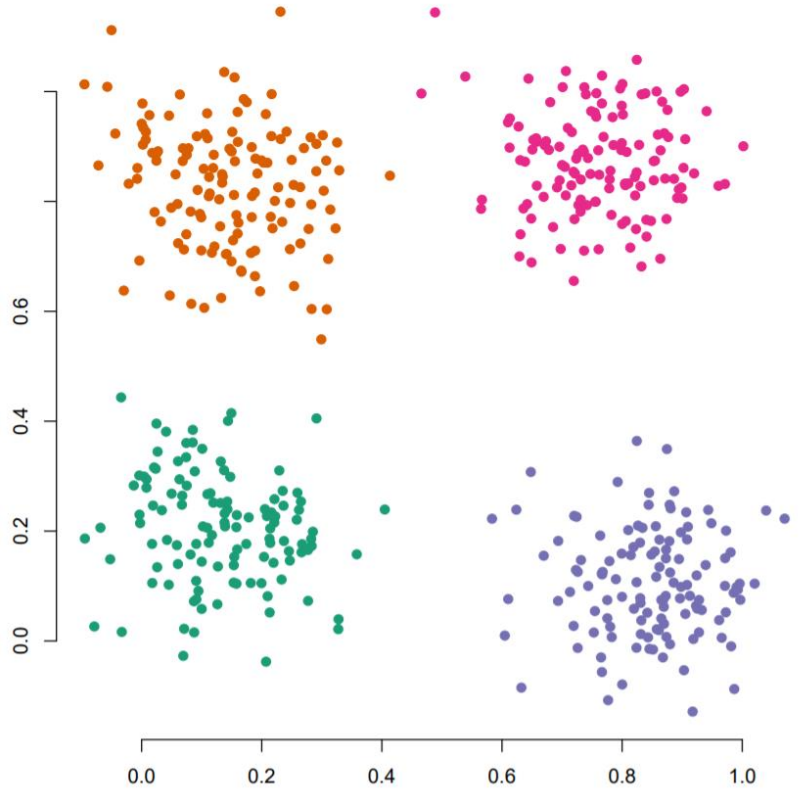
- What is a good distance function here?
- Squared Euclidean distance is reasonable
- $d(x_n, x_m) = \|x_n - x_m\|^2$

Example data



- Goal: segment this data into k groups
- What should k be?
- Automatically choosing k is complicated; for now, 4

K-means



- Different clustering algorithms use the data and distance measurements in different ways
- Begin with k-means, the simplest clustering algorithm

K-means

- The basic idea is to describe each cluster by its mean value
- This works only for distances such that a mean is well-defined
- The goal of k-means is to assign data to clusters and define these clusters with their means

K-means algorithm

- Initialization
  - Data: $x_{1:N}$
  - Choose initial cluster means $m_{1:k}$ (same dimension as data)
- Repeat
  - Assign each data point to its closest mean

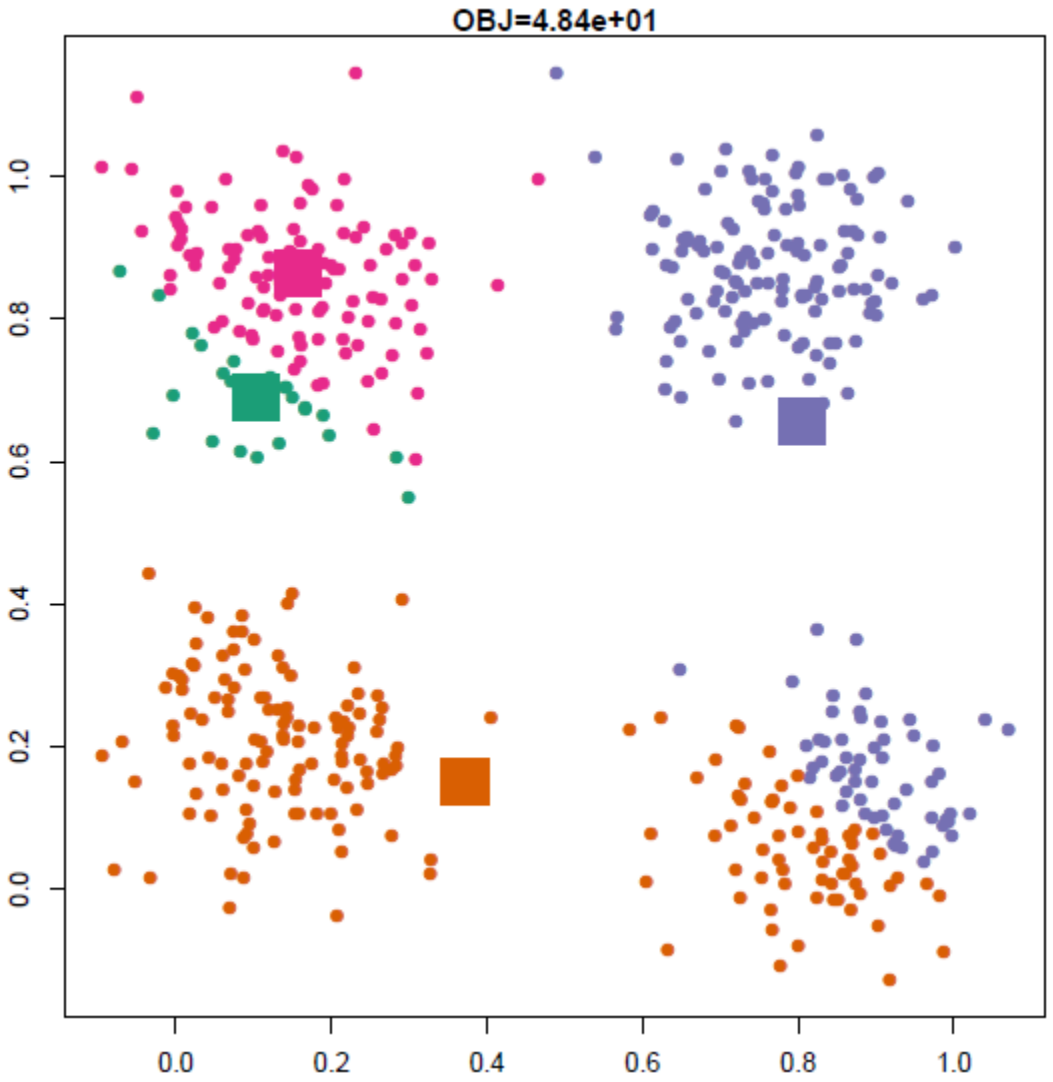
$$z_n = \arg \min_{i \in \{1,\ldots,k\}} d(\mathbf{x}_n, \mathbf{m}_i)$$

  - Compute each cluster mean to be the coordinate-wise average over data points assigned to that cluster,

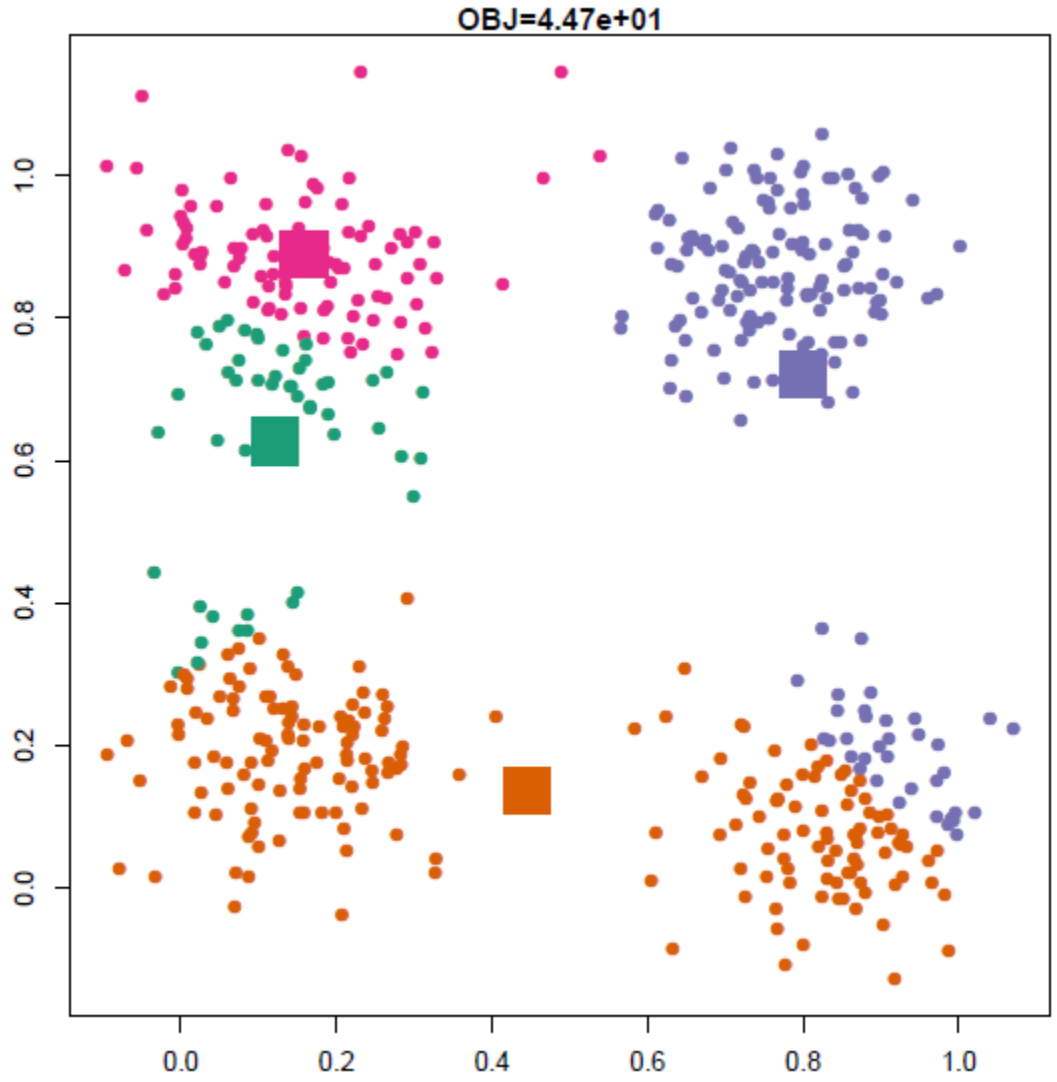
$$\mathbf{m}_k = \frac{1}{N_k} \sum_{\{n : z_n = k\}} \mathbf{x}_n$$

- Until assignments $z_{1:N}$ do not change

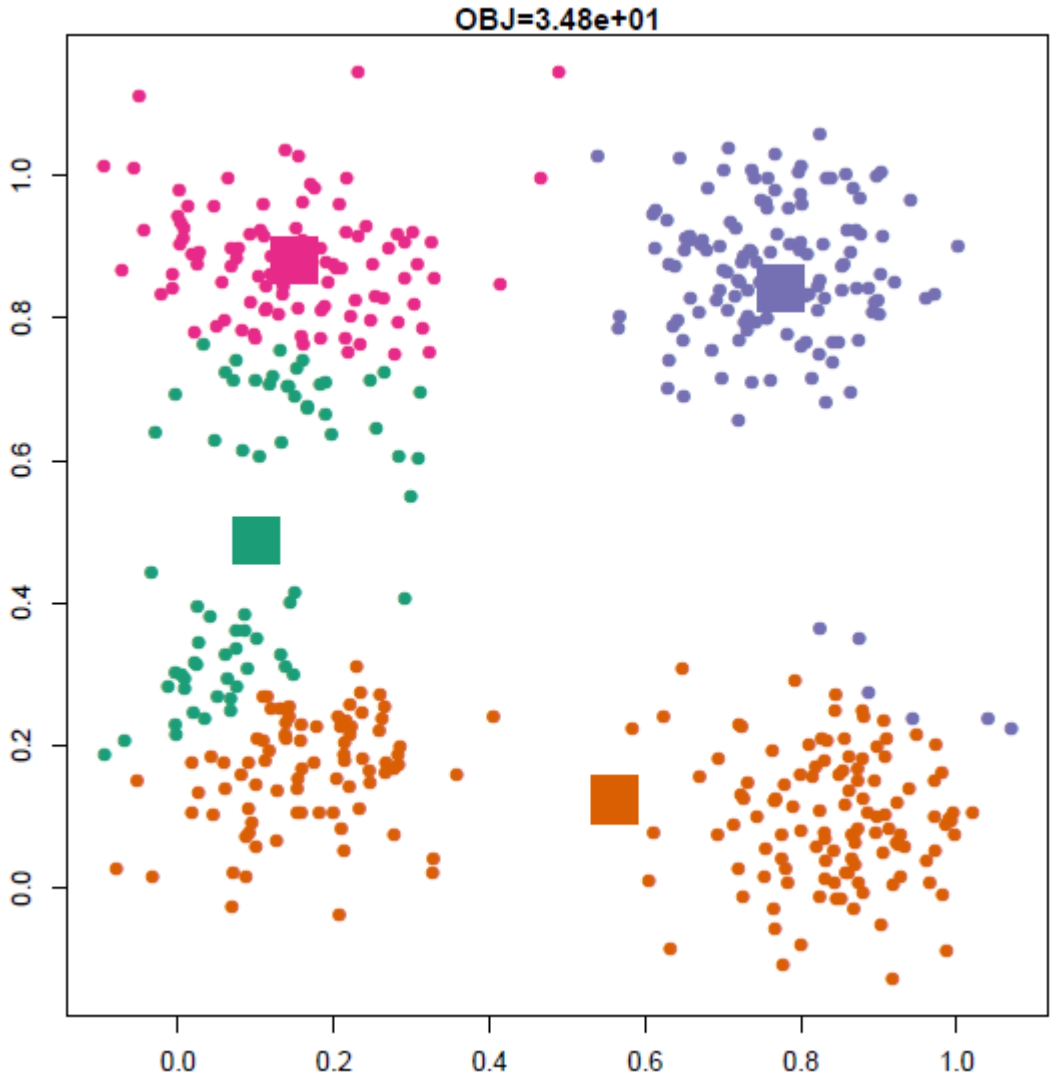
K-means example



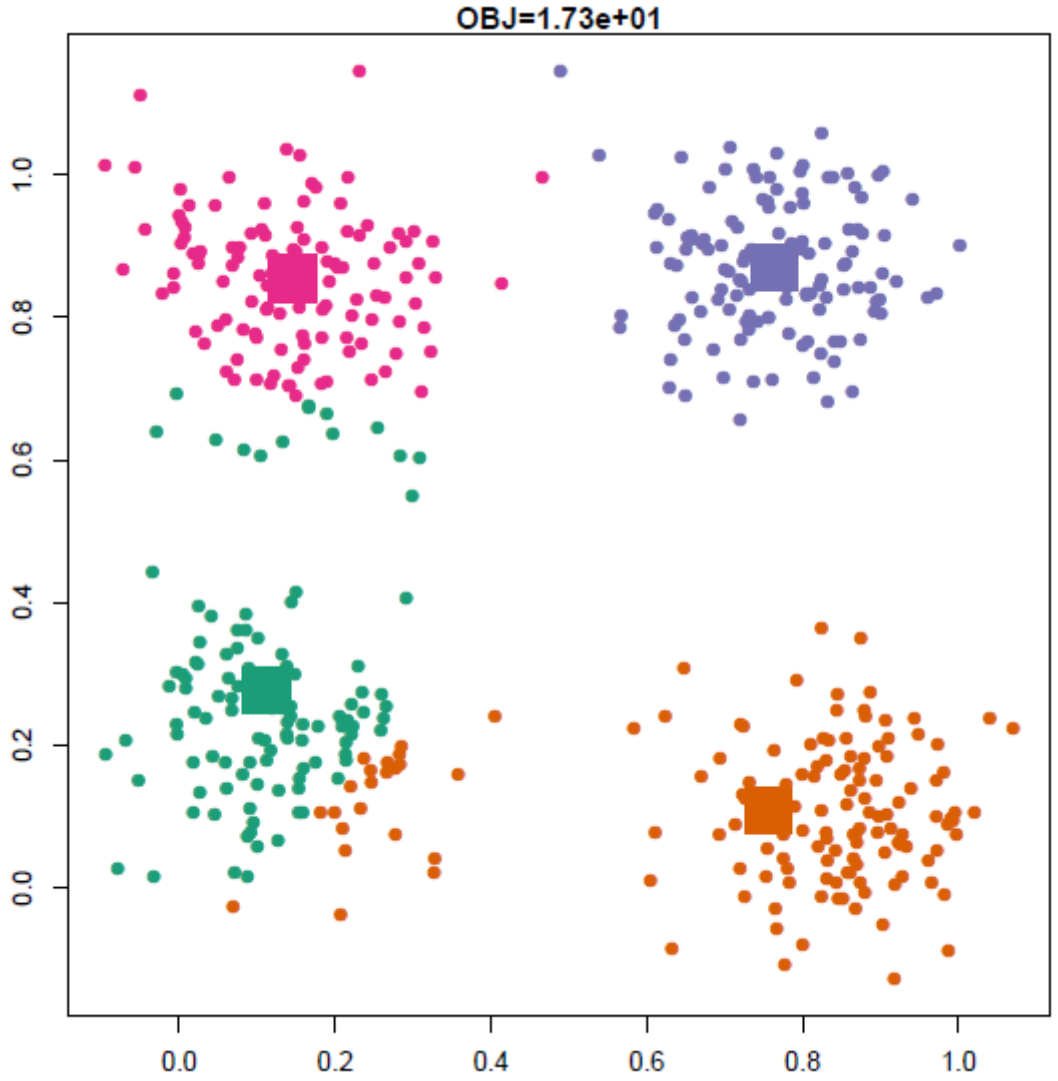
K-means example



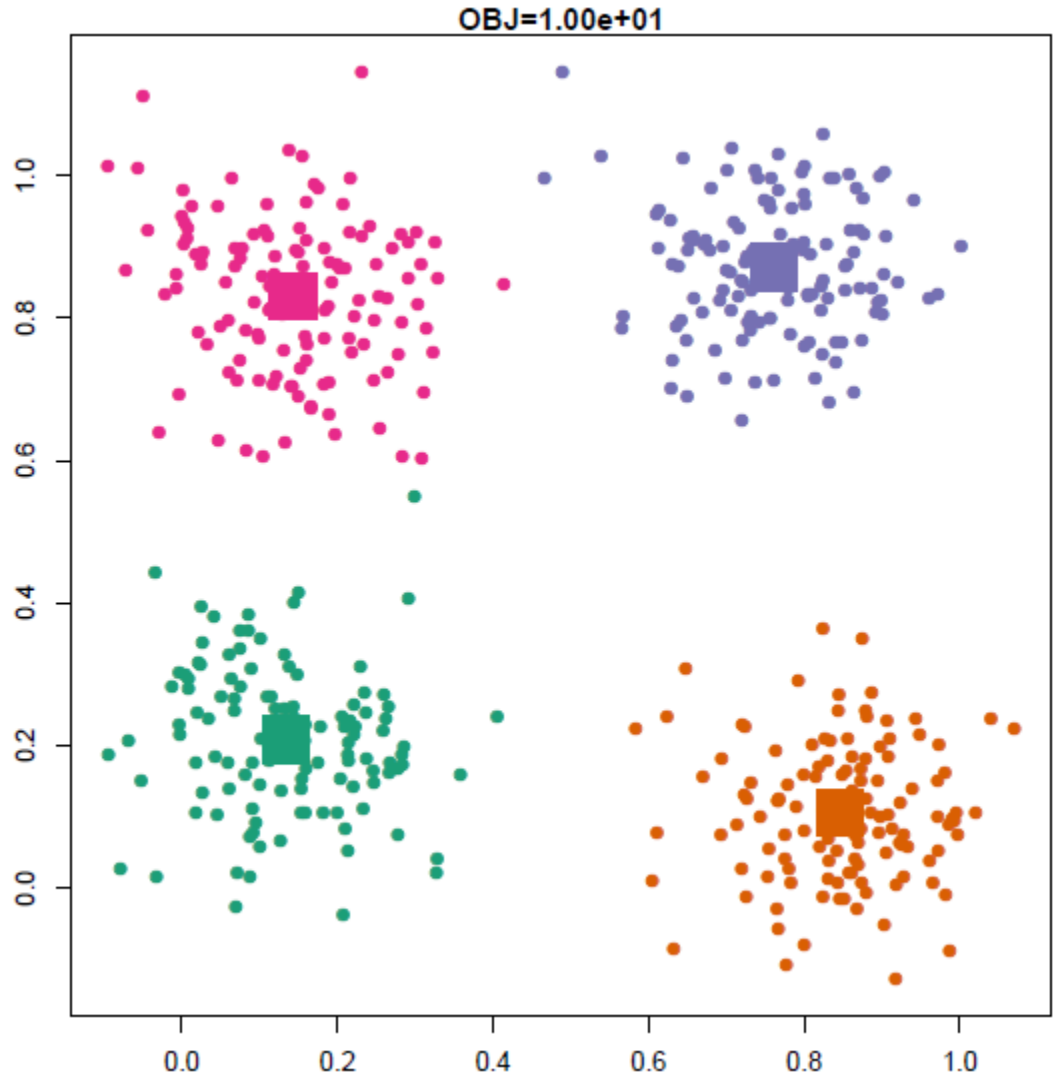
K-means example



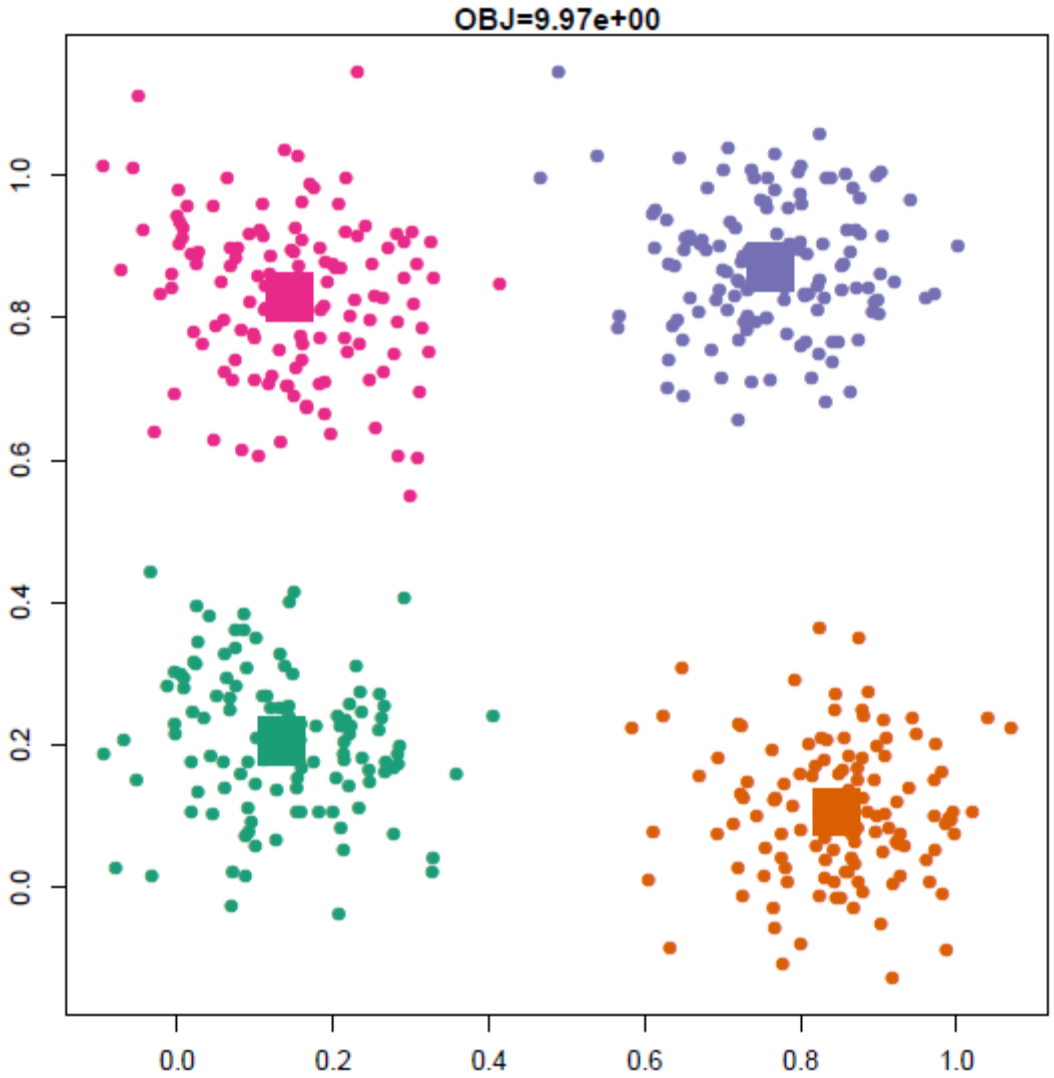
K-means example



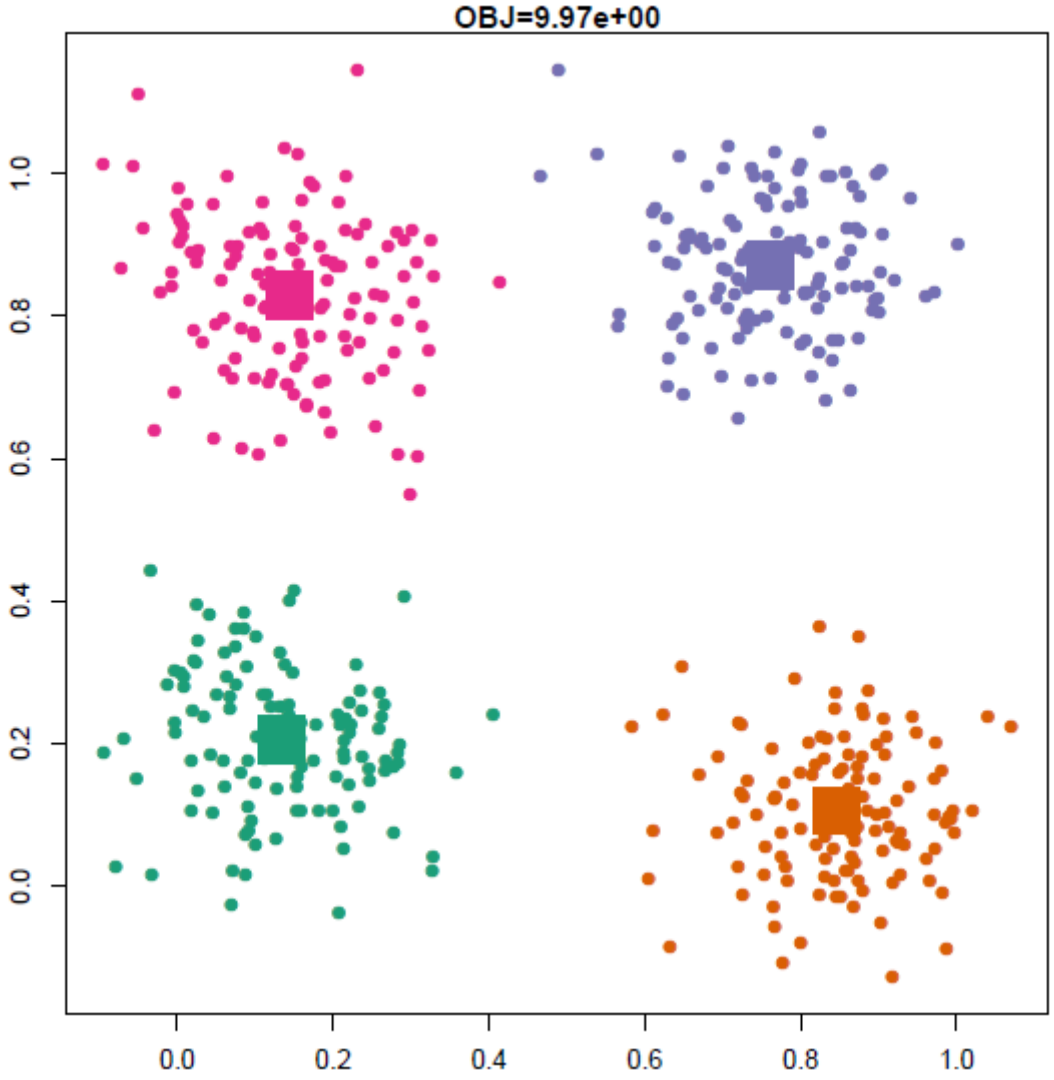
K-means example



K-means example



K-means example (converges)



Objective function

- How can we measure how well our algorithm is doing?

- The k-means objective function is the sum of the squared distances of each point to each assigned mean

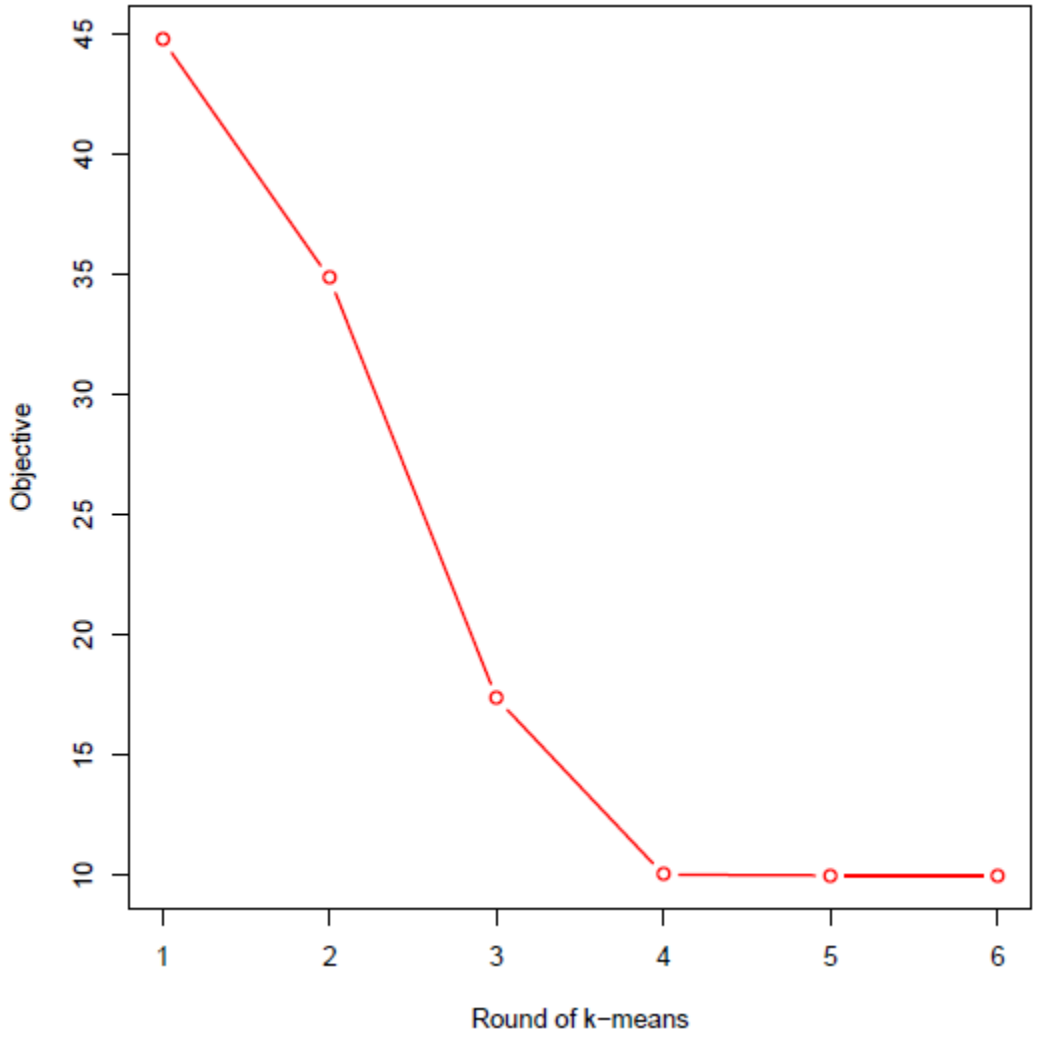
$$F(z_{1:N}, \mathbf{m}_{1:k}) = \frac{1}{2} \sum_{n=1}^{N} ||\mathbf{x}_n - \mathbf{m}_{z_n}||^2$$

Coordinate Descent

$$F(z_{1:N}, \mathbf{m}_{1:k}) = \frac{1}{2} \sum_{n=1}^{N} ||\mathbf{x}_n - \mathbf{m}_{z_n}||^2$$

- Holding the means fixed, assigning each point to its closest mean minimizes F with respect to $z_{1:N}$.
- Holding the assignments fixed, computing the centroids of each cluster minimizes F with respect to $m_{1:k}$.
- Thus, k-means is a coordinate descent algorithm.
- It finds a local minimum. (Multiple restarts are often necessary.)

Objective for the example data



K-medoids

- In many practical settings, Euclidean distance is not appropriate
- For example,
  - Discrete multivariate data, such as purchase histories
  - Positive data, such as time spent on a web-page
- k-medoids is an algorithm that only requires knowing distances between data points, $d_{n,m} = d(x_n, x_{mk})$
- *No need to define the mean*
- Each of the clusters is associated with its most typical example

K-medoids algorithm

- Initialization
  - Data: $x_{1:N}$
  - Choose initial cluster identities $m_{1:k}$
- Repeat
  - Assign each data point to its closest center

$$z_n = \arg \min_{i \in \{1,\ldots,k\}} d(\mathbf{x}_n, \mathbf{m}_i)$$

  - For each cluster, find the data point in that cluster that is closest to the other points in that cluster

$$i_k = \arg \min_{\{n : z_n = k\}} \sum_{\{m : z_m = k\}} d(\mathbf{x}_n, \mathbf{x}_m)$$

  - Set each cluster center equal to their closest data points

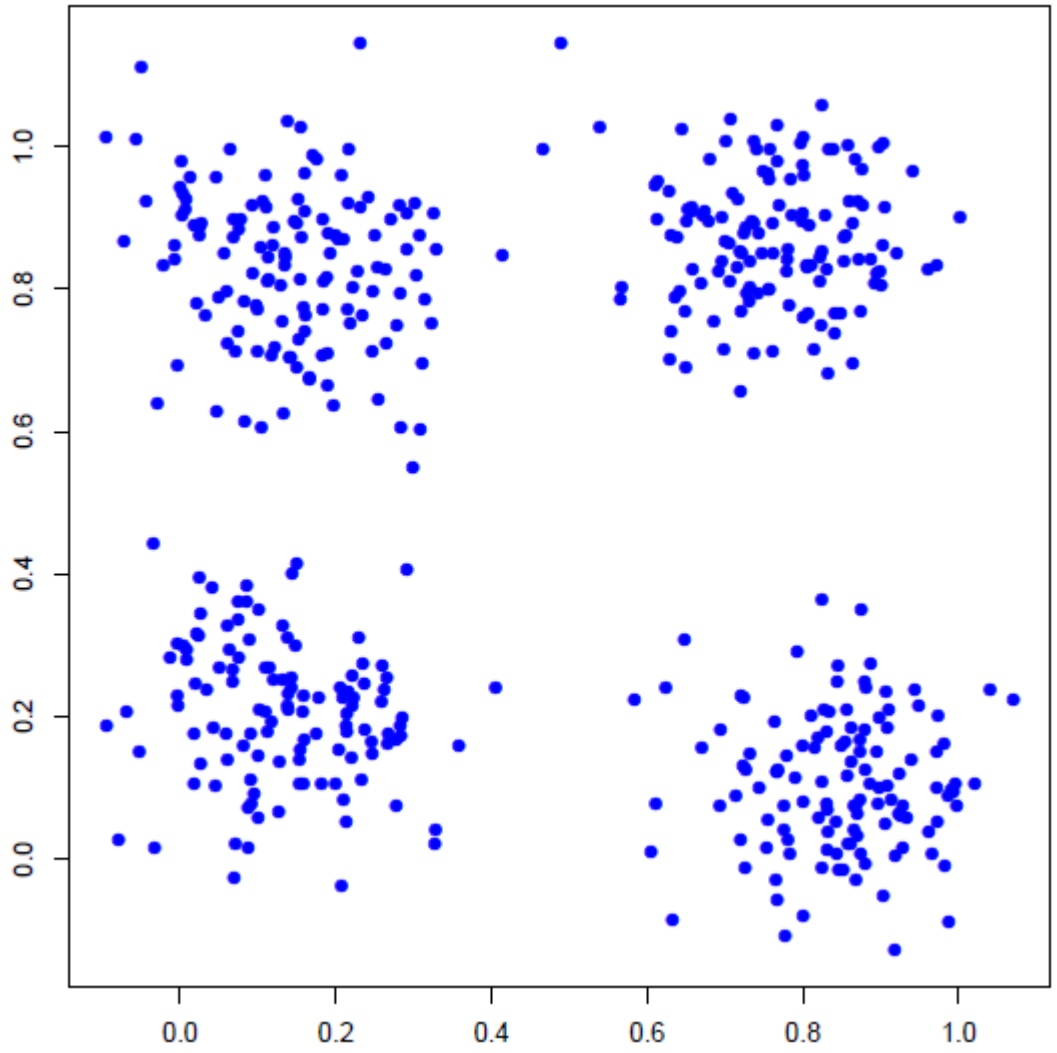
$$m_k = \mathbf{x}_{i_k}$$

- Until assignments z1:N do not change

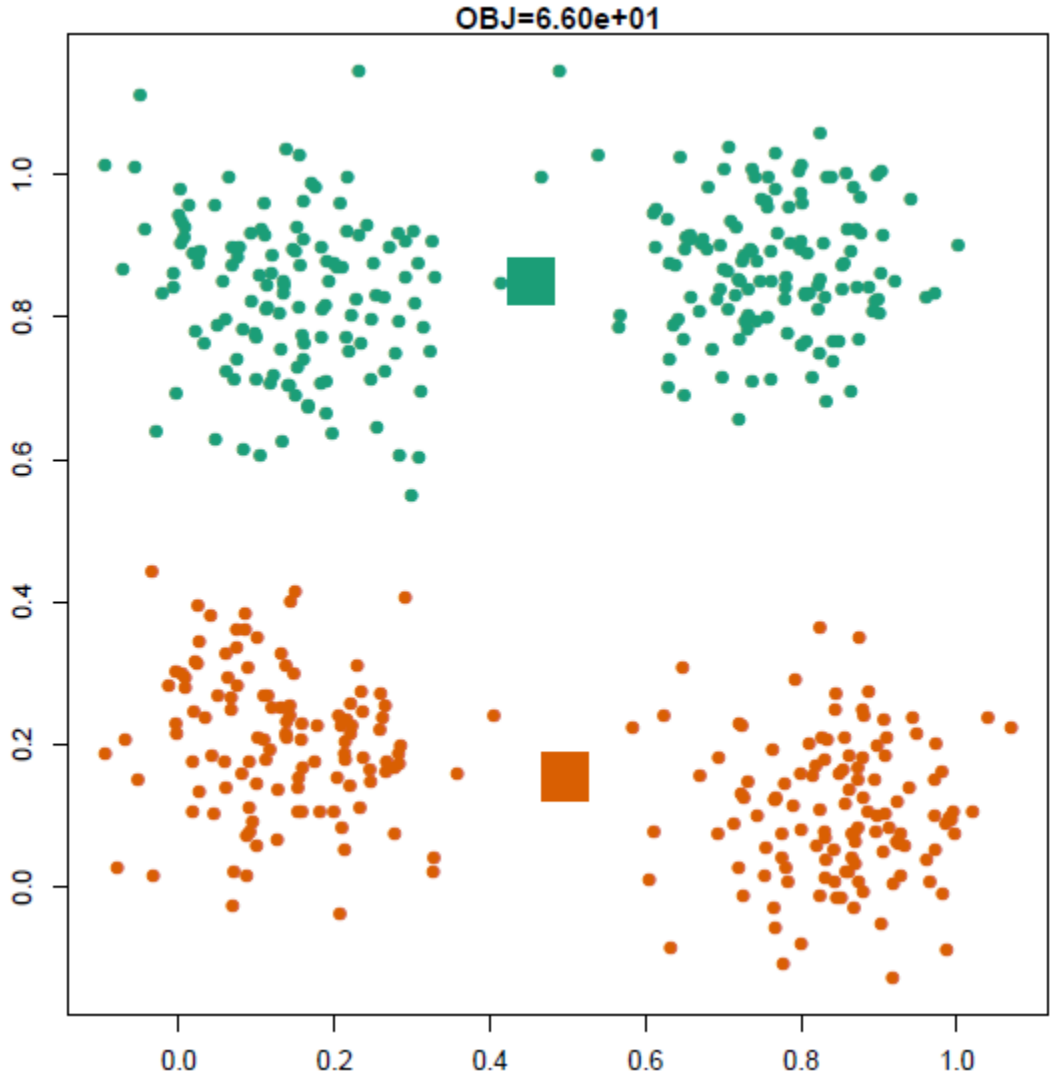
Choosing *k*

- Choosing *k* is a nagging problem in cluster analysis
- Sometimes, the problem determines *k*
  - Example: Clustering customers for k salespeople in a business
- Usually, we seek the “natural” clustering, but what does this mean?
- It is not well-defined

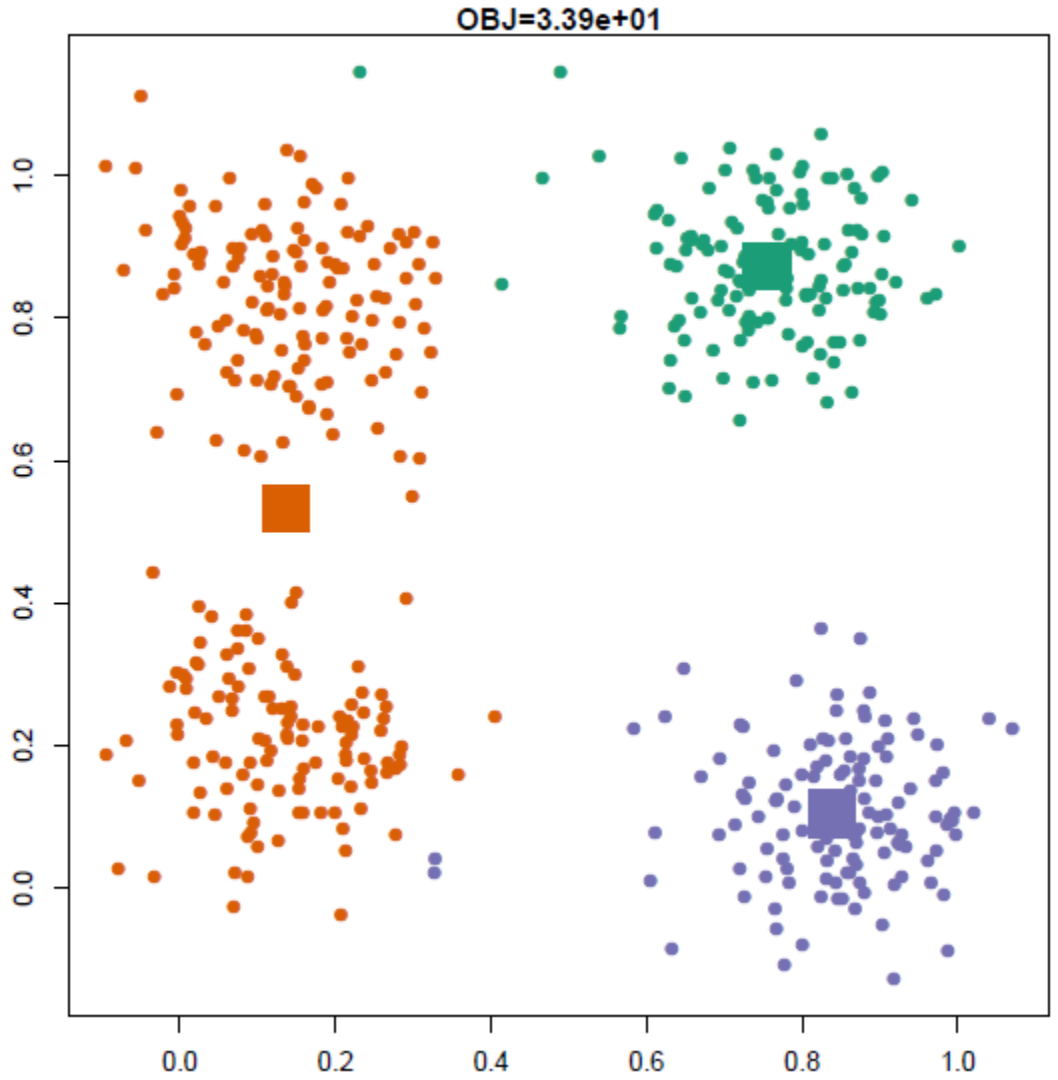
What happens as *k* increases?



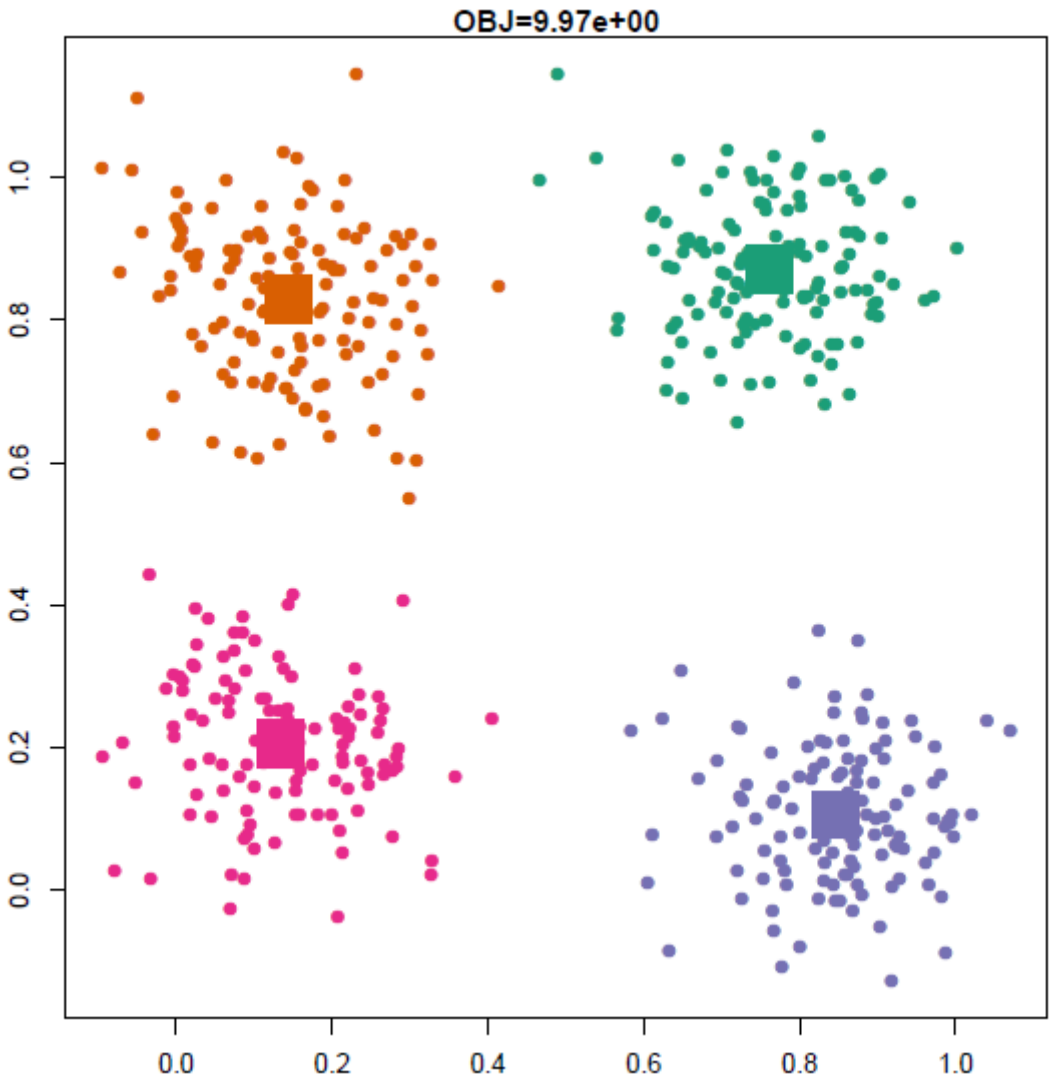
What happens as *k* increases?



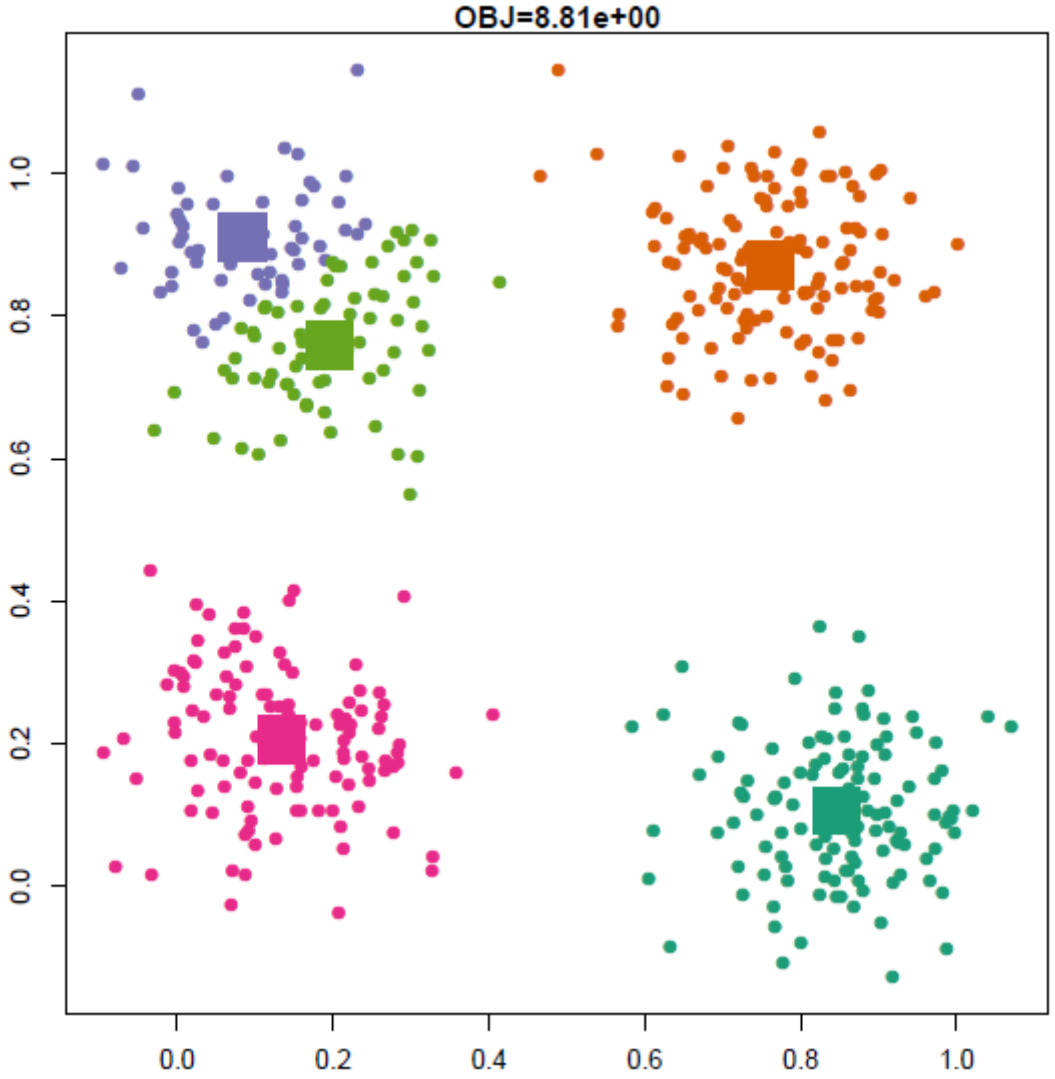
What happens as *k* increases?



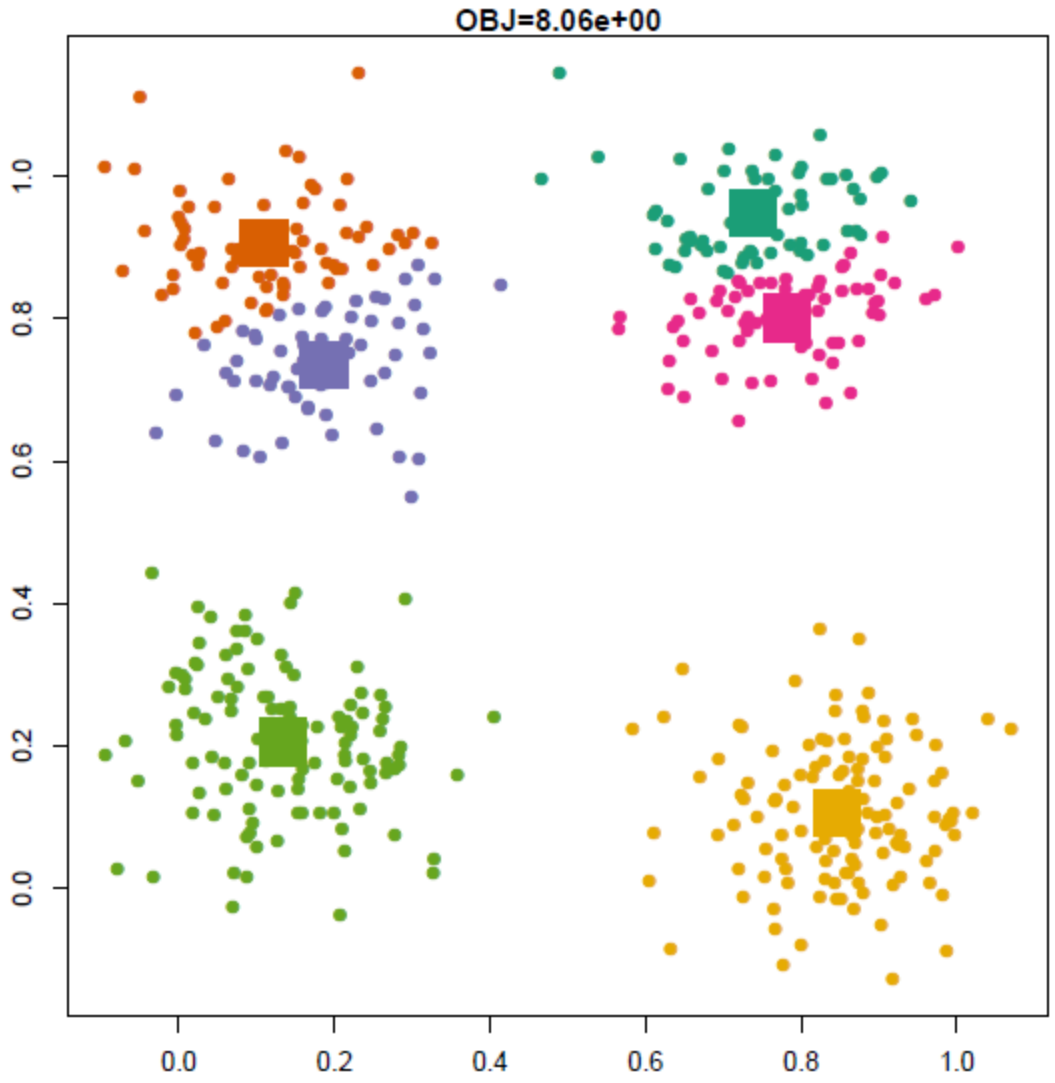
What happens as *k* increases?



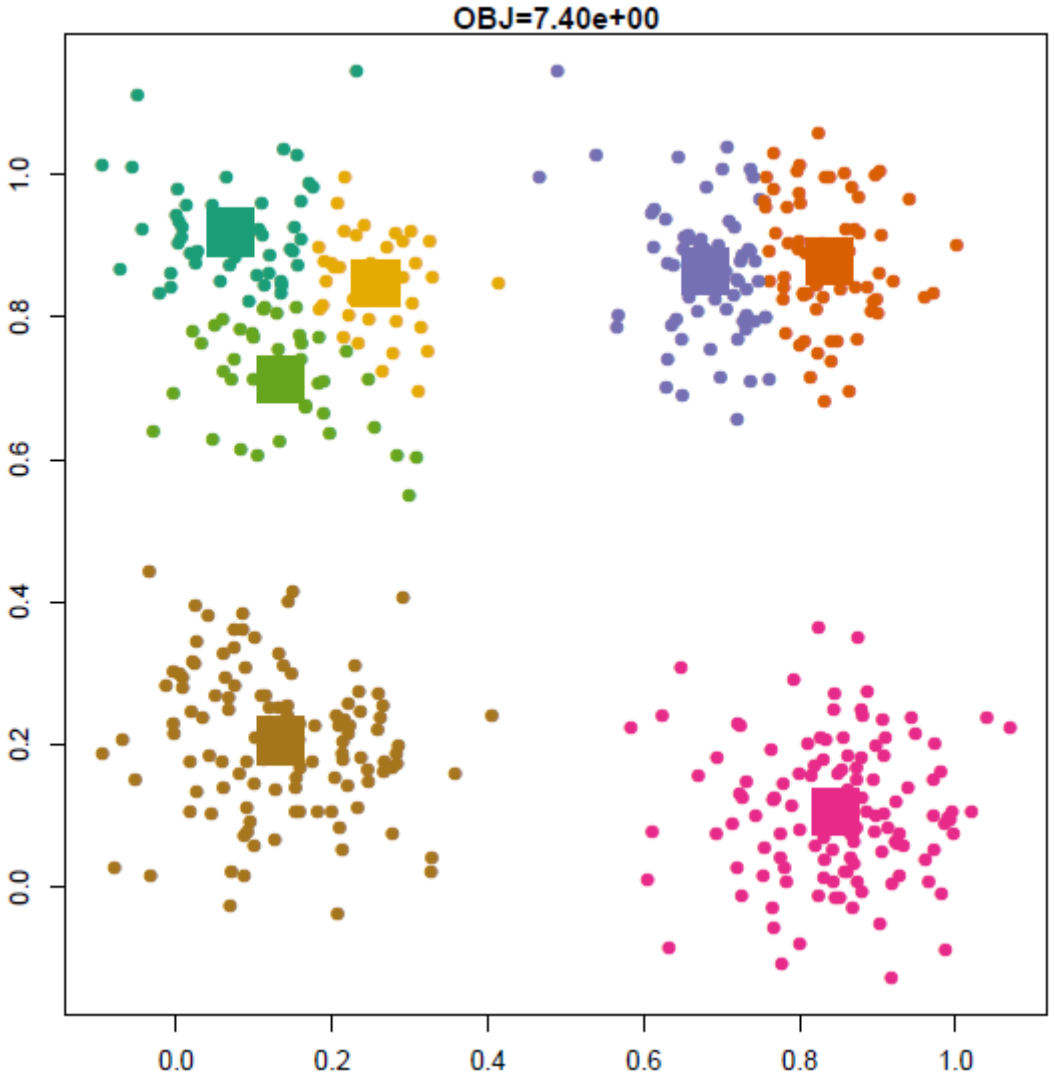
What happens as *k* increases?



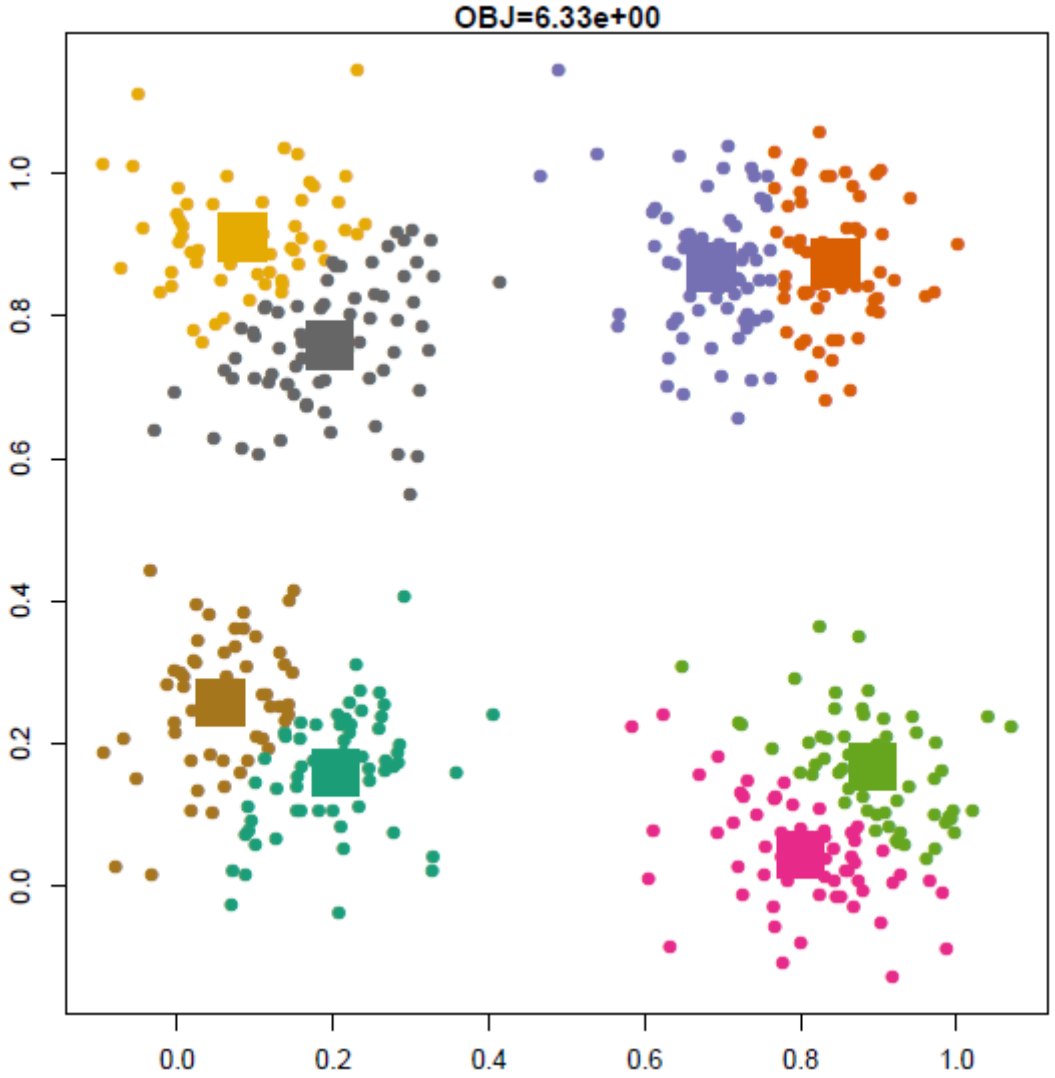
What happens as *k* increases?



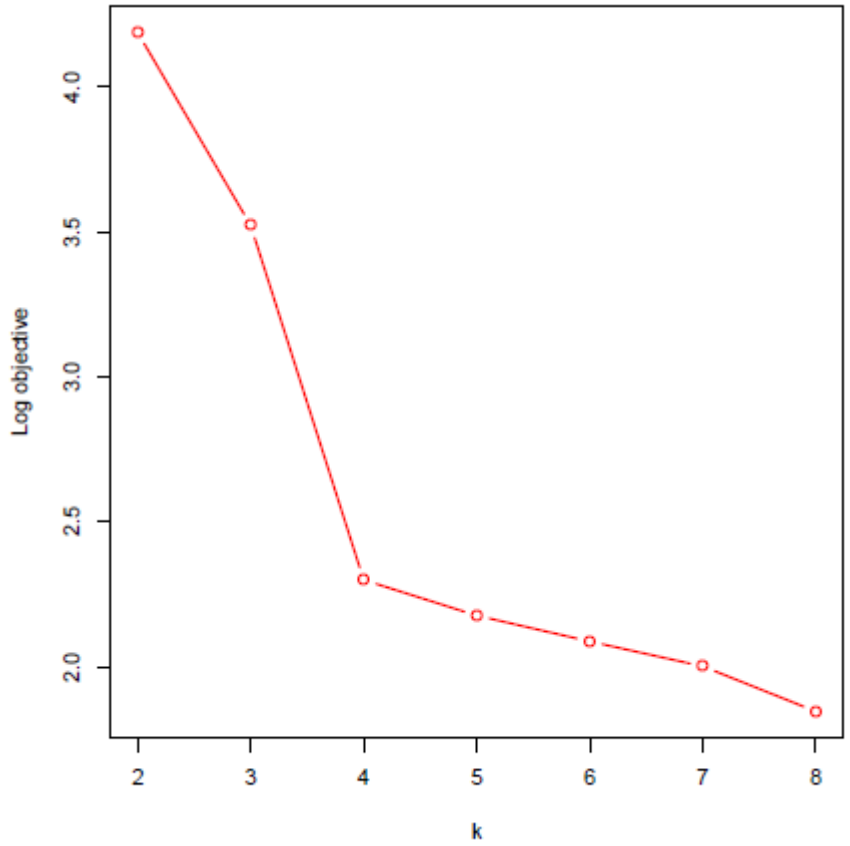
What happens as *k* increases?



What happens as *k* increases?



Heuristic: A kink in the objective



- Notice the “kink” in the objective between 3 and 5
- This suggests that 4 is the right number of clusters

Hierarchical Clustering

Hierarchical Clustering

- Hierarchical clustering is a widely used data analysis tool
- The idea is to build a binary tree of the data that successively merges similar groups of points
- Visualizing this tree provides a useful summary of the data

Hierarchical clustering vs. k-means

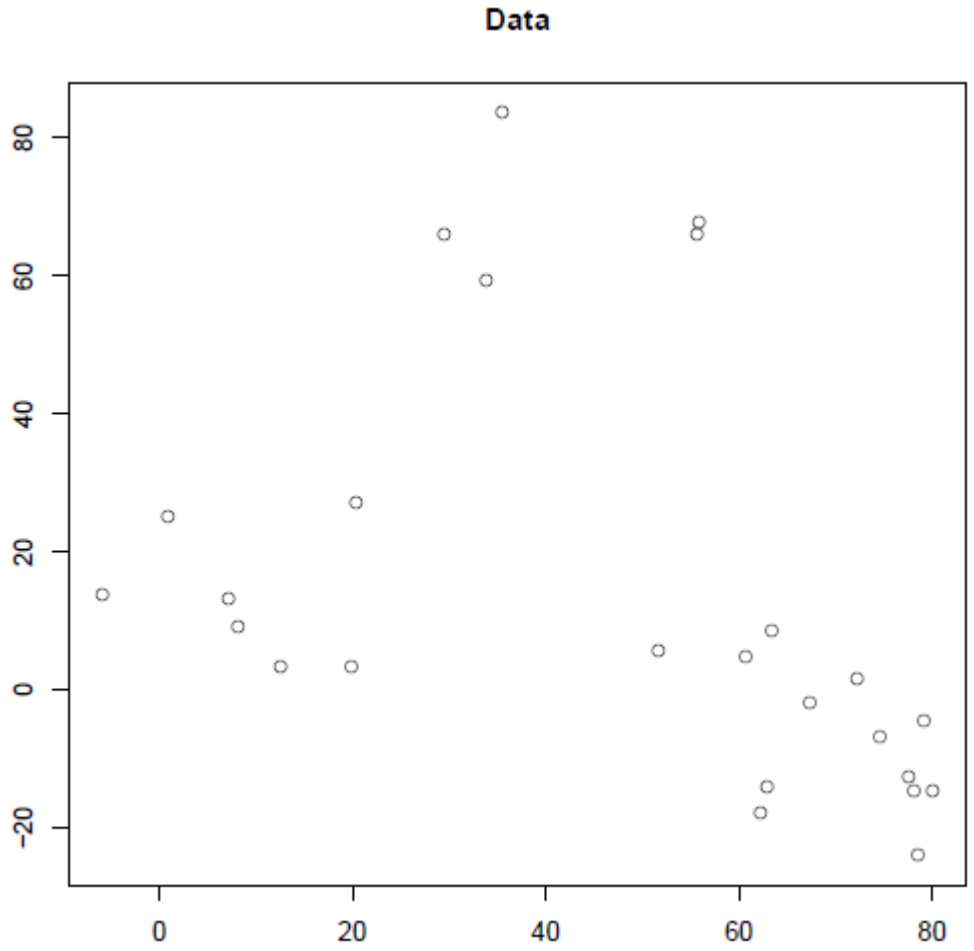
- k-means or k-medoids requires
  - number of clusters k
  - initial assignment of data to clusters
  - distance measure between data $d(x_n, x_m)$
- Hierarchical clustering only requires a measure of similarity between groups of data points.

Agglomerative clustering

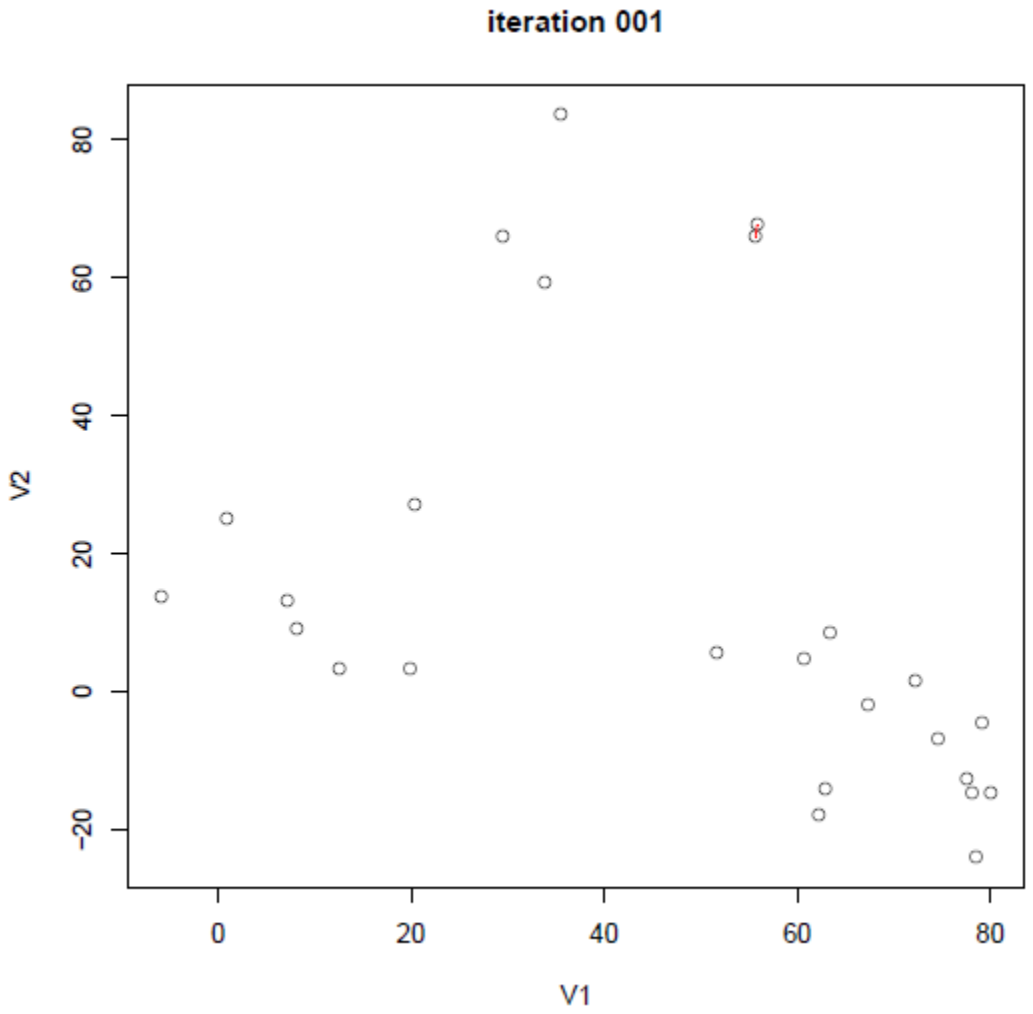
Algorithm:

1. Place each data point into its own singleton group
2. Repeat: iteratively merge the two closest groups
3. Until: all the data are merged into a single cluster

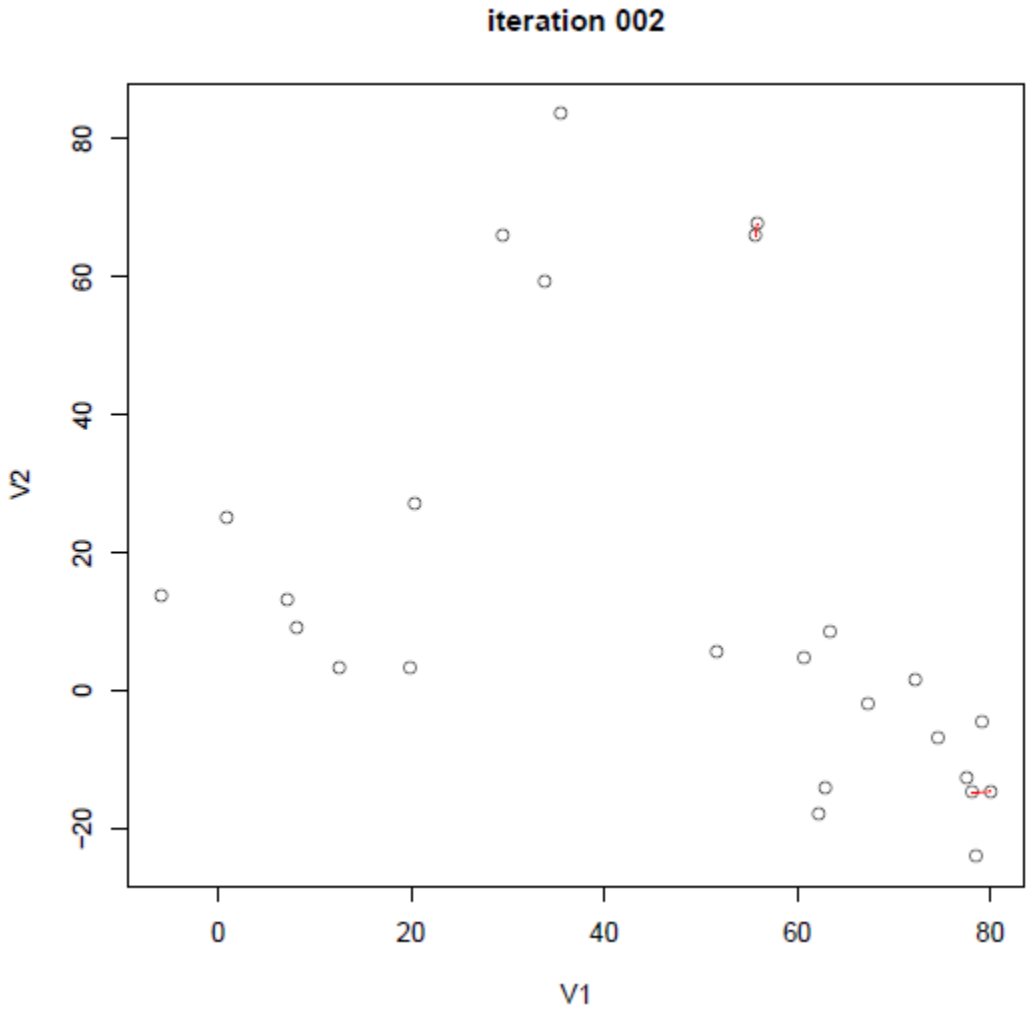
Example



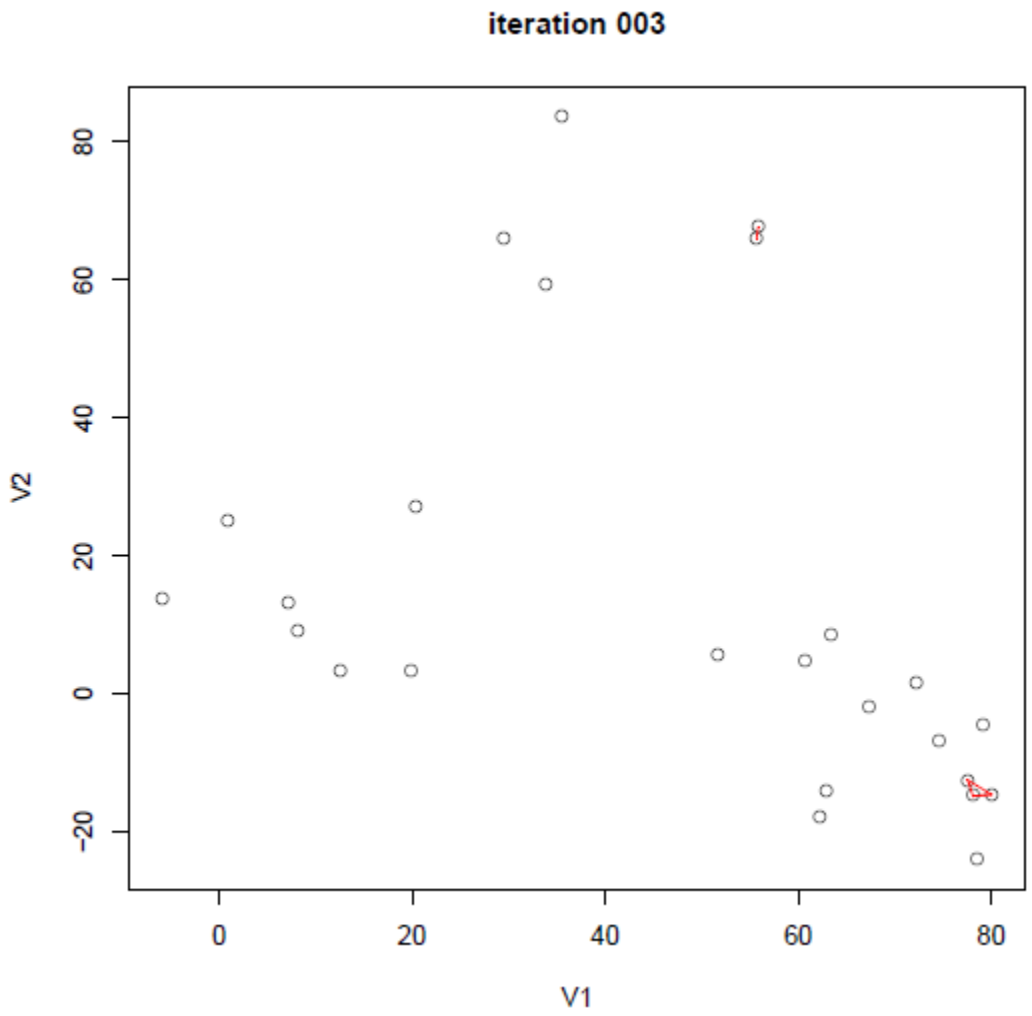
Example



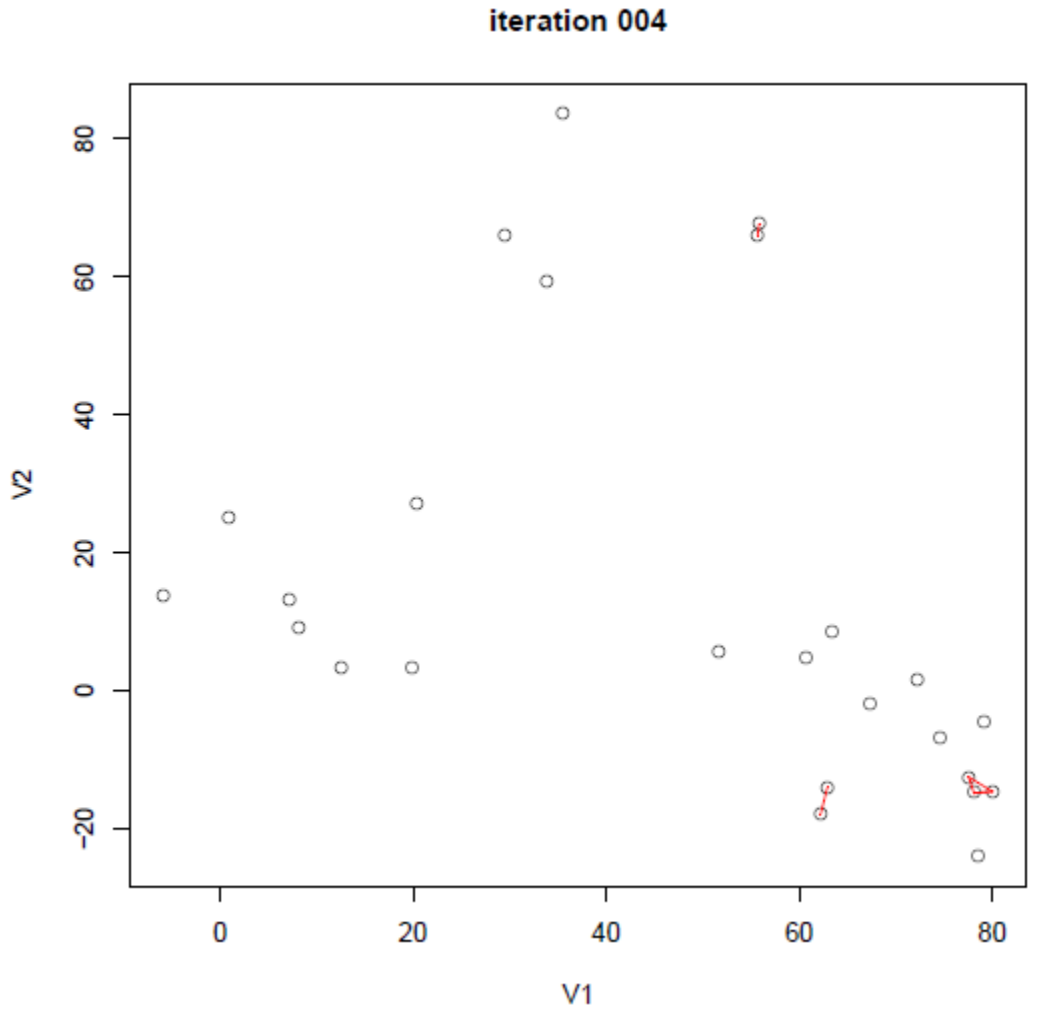
Example



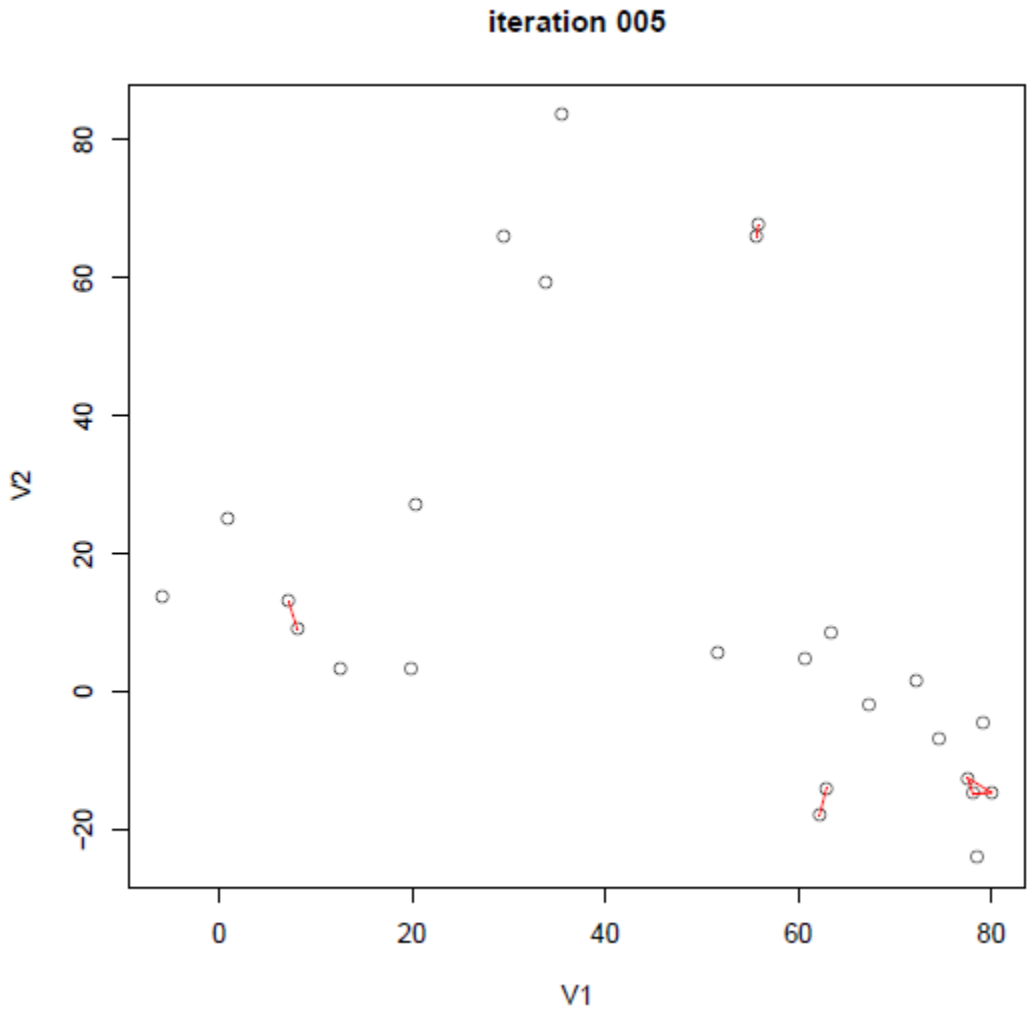
Example



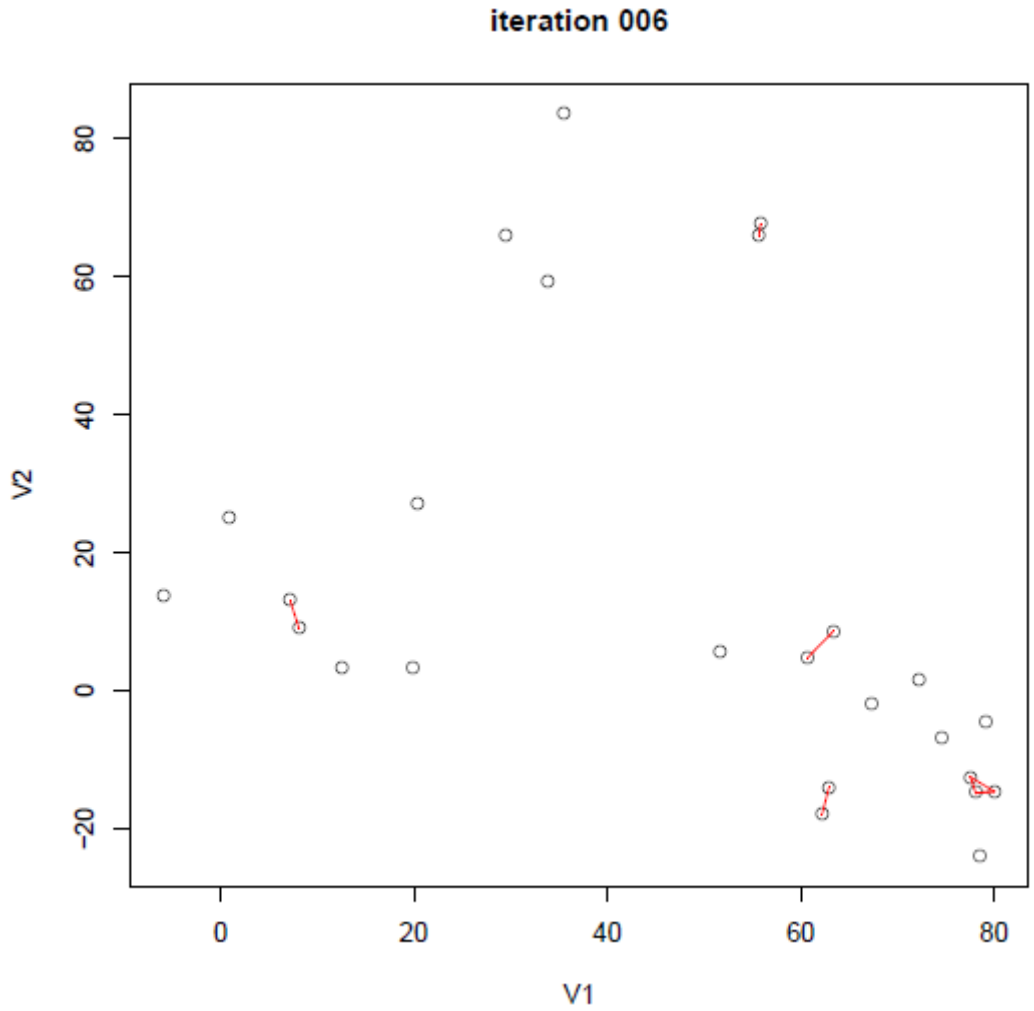
Example



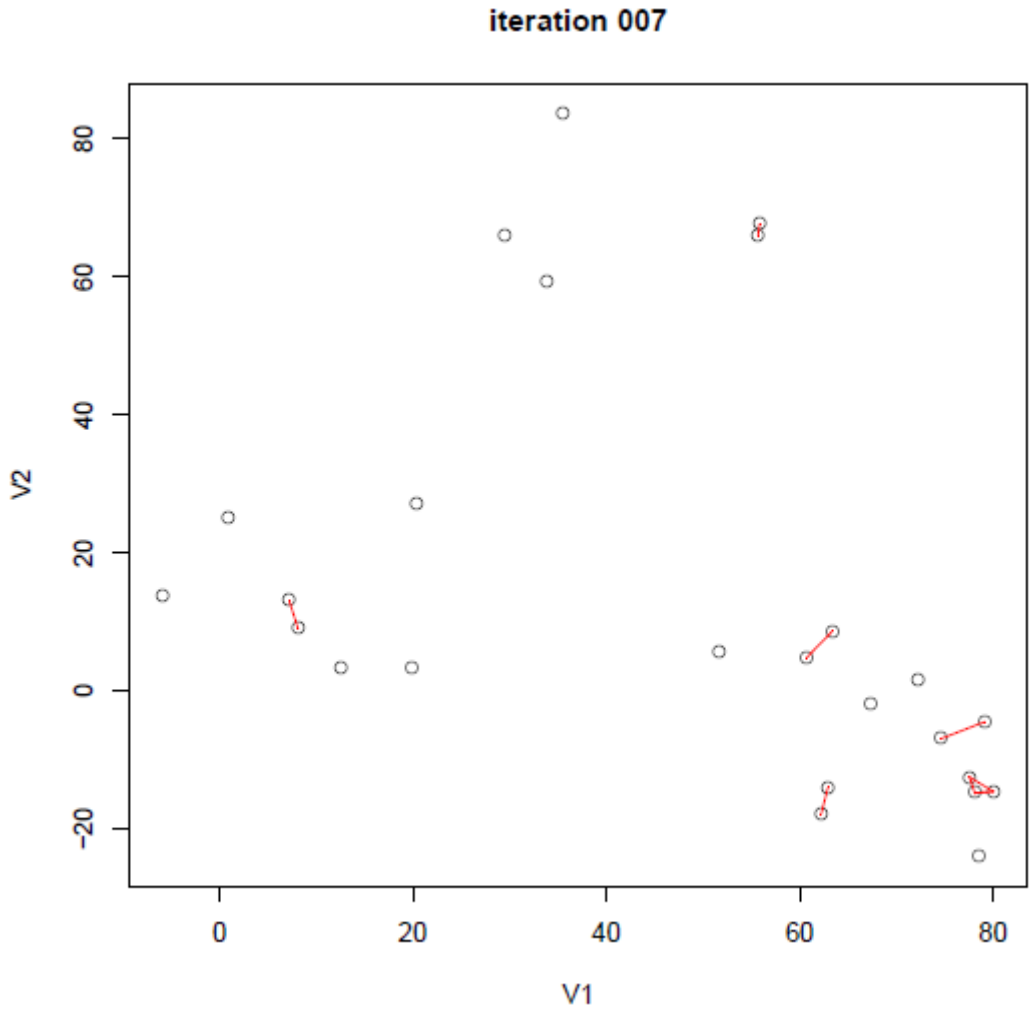
Example



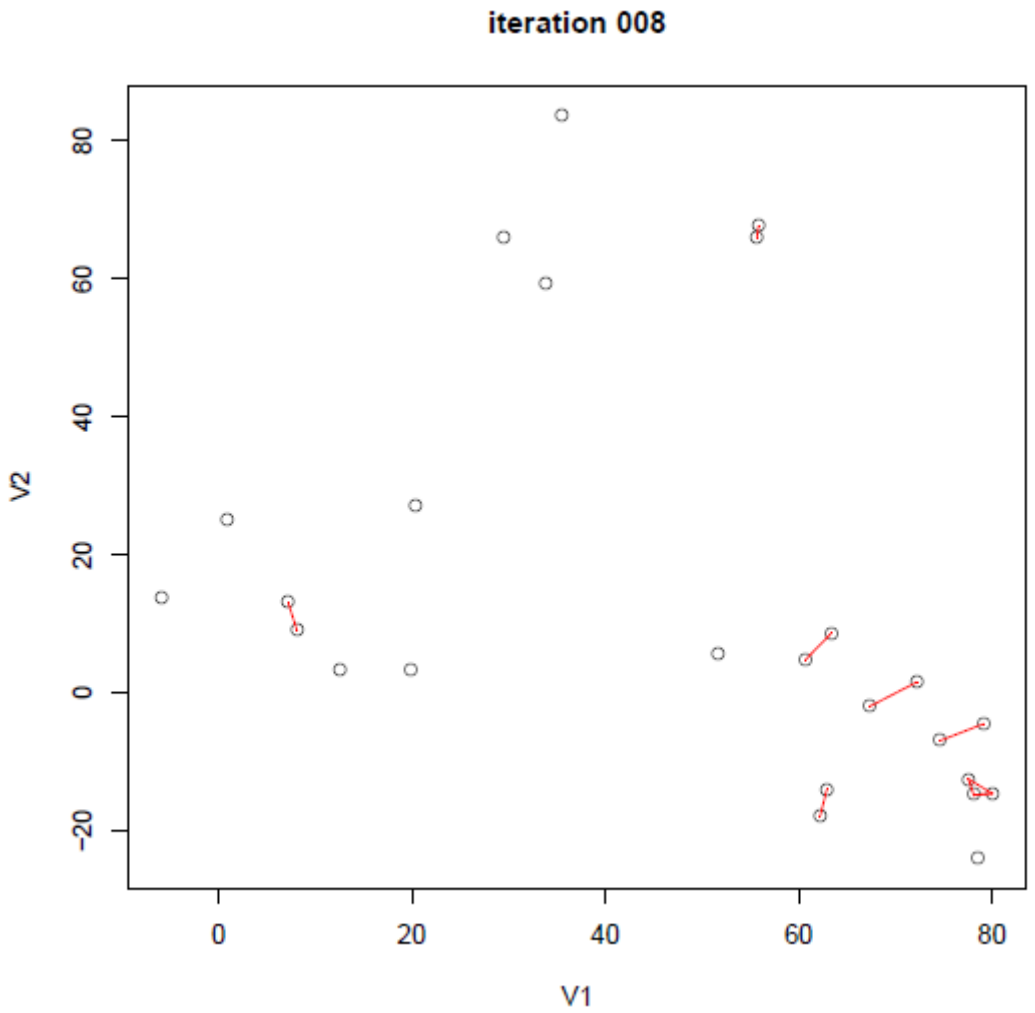
Example



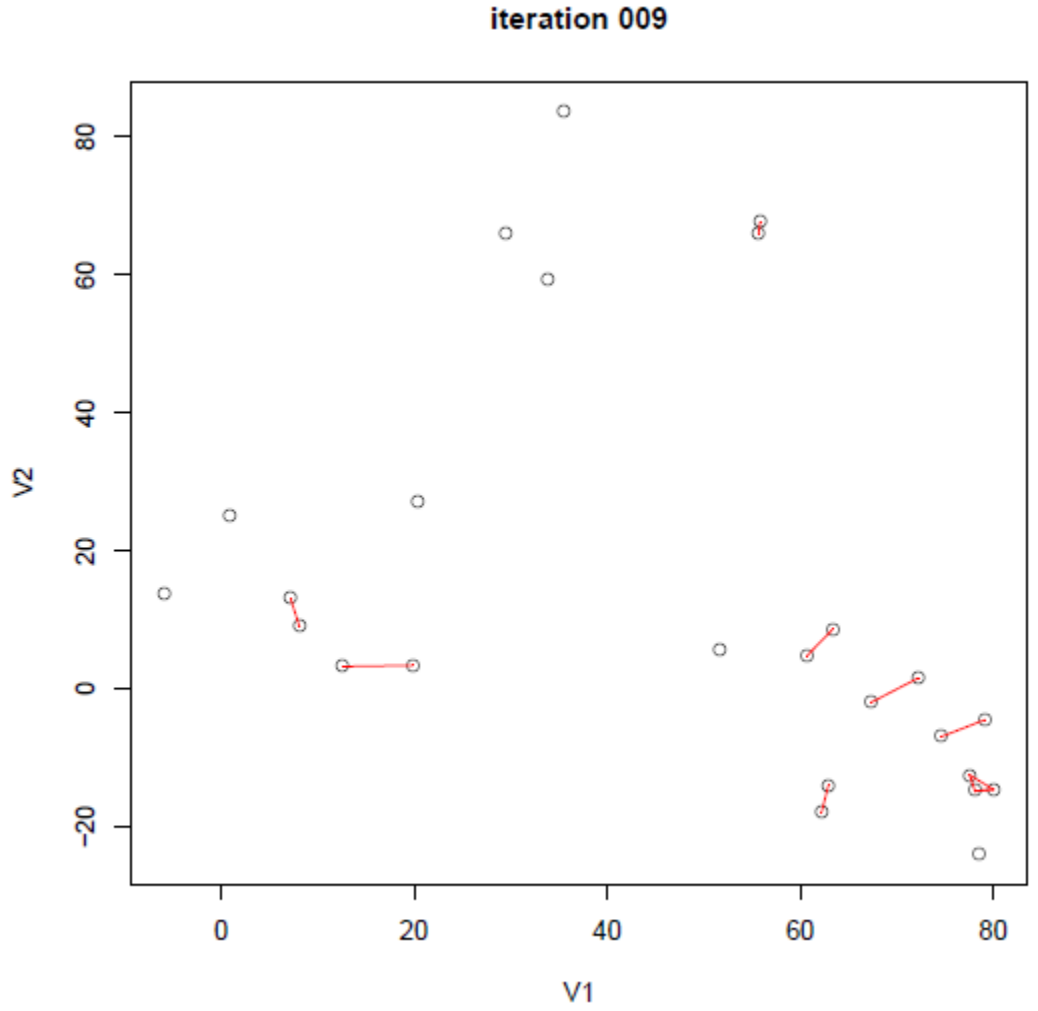
Example



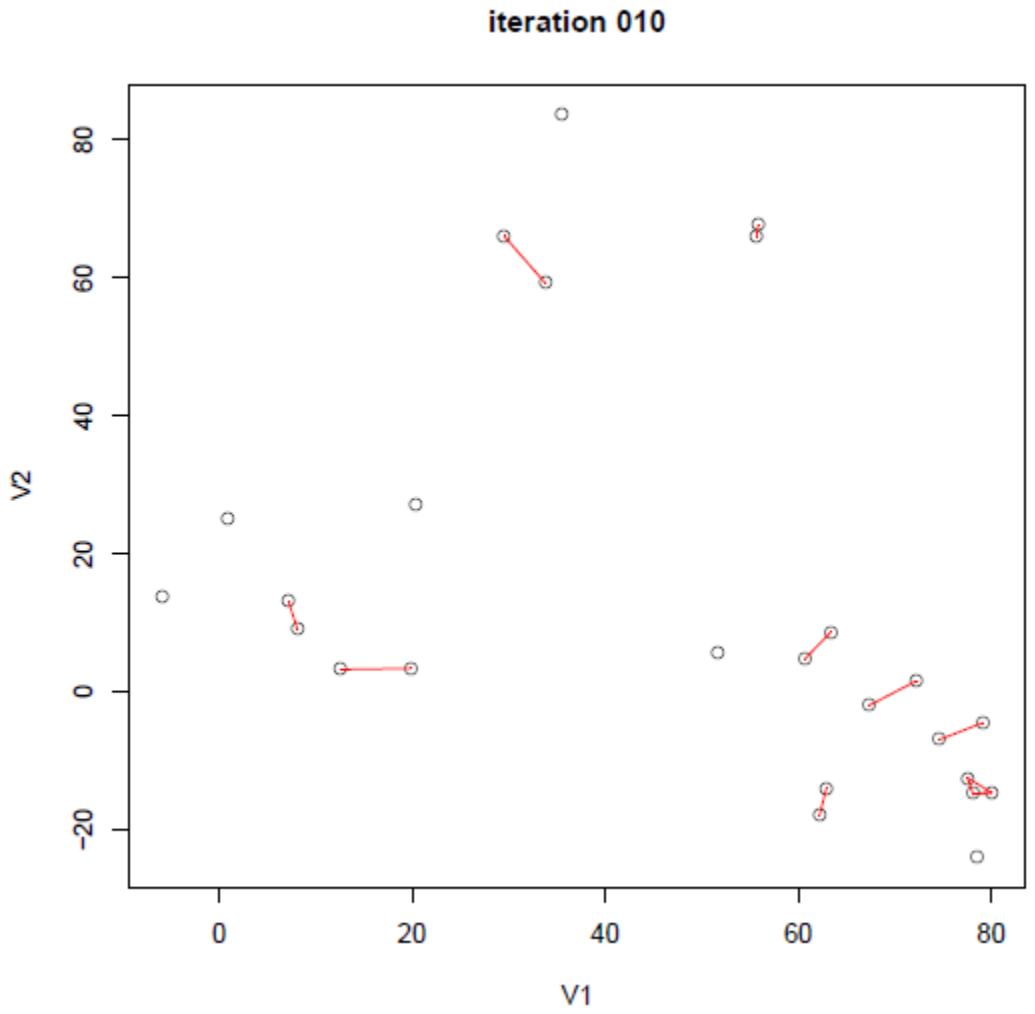
Example



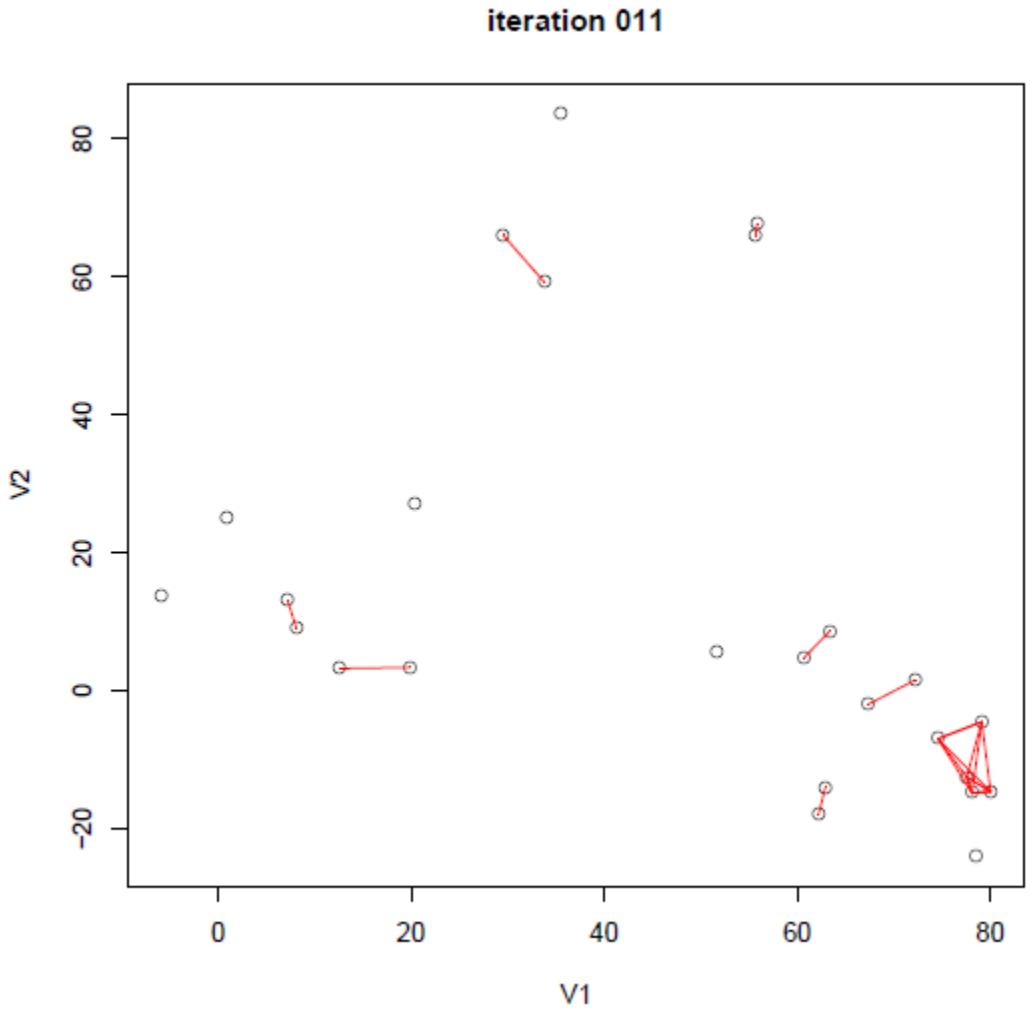
Example



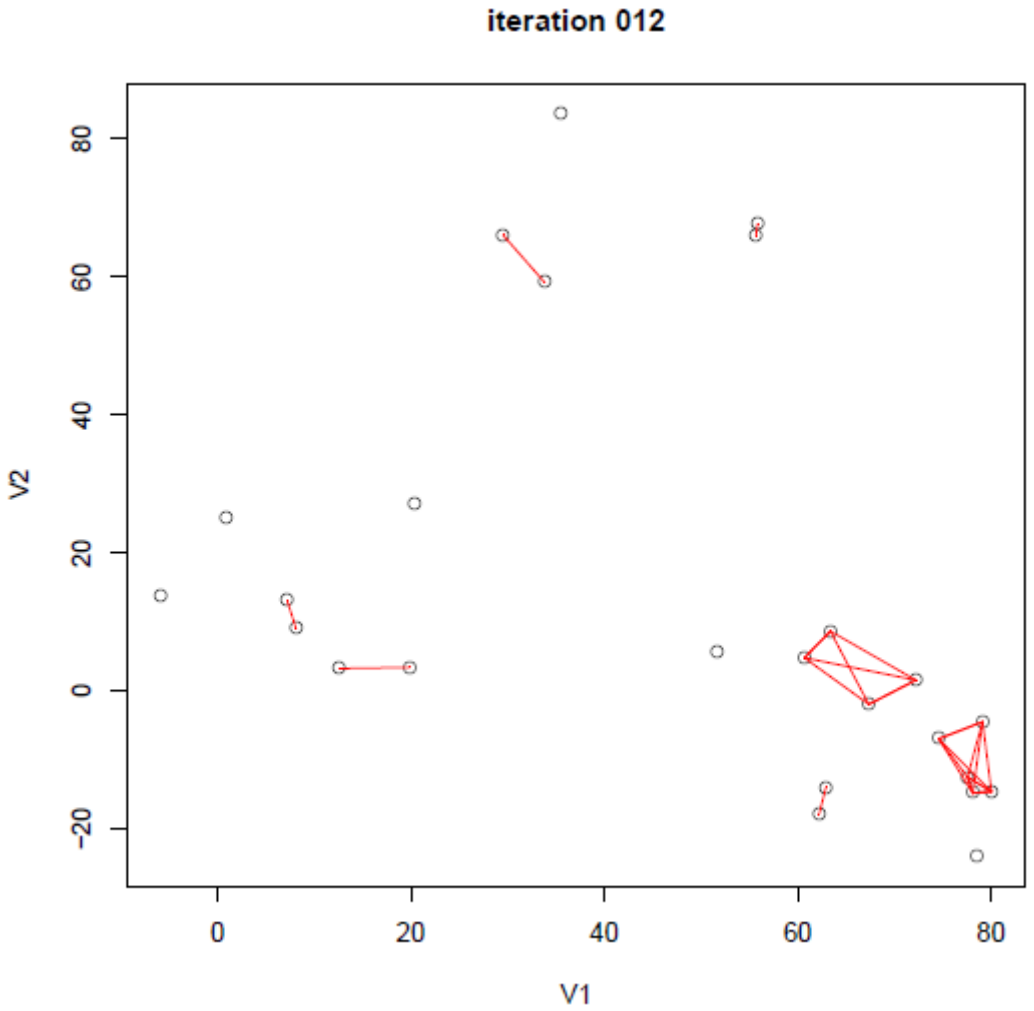
Example



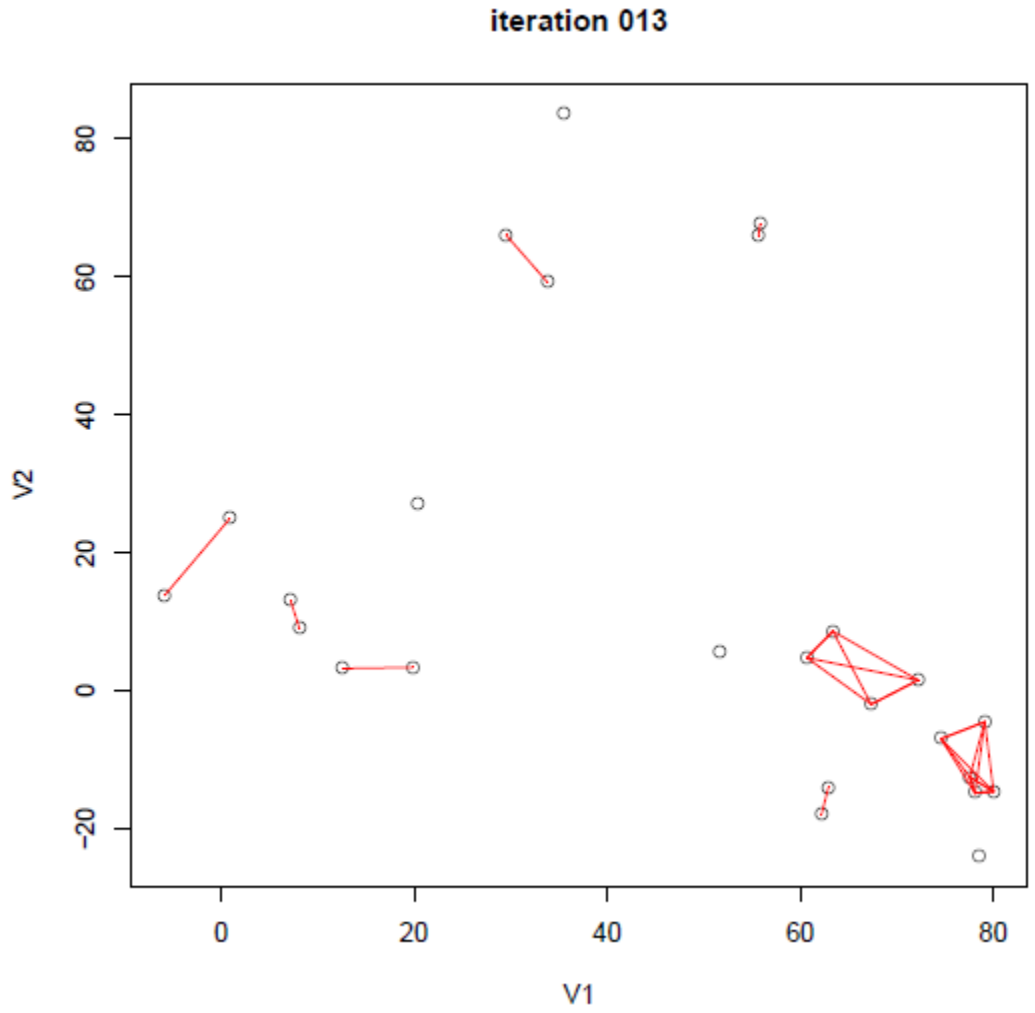
Example



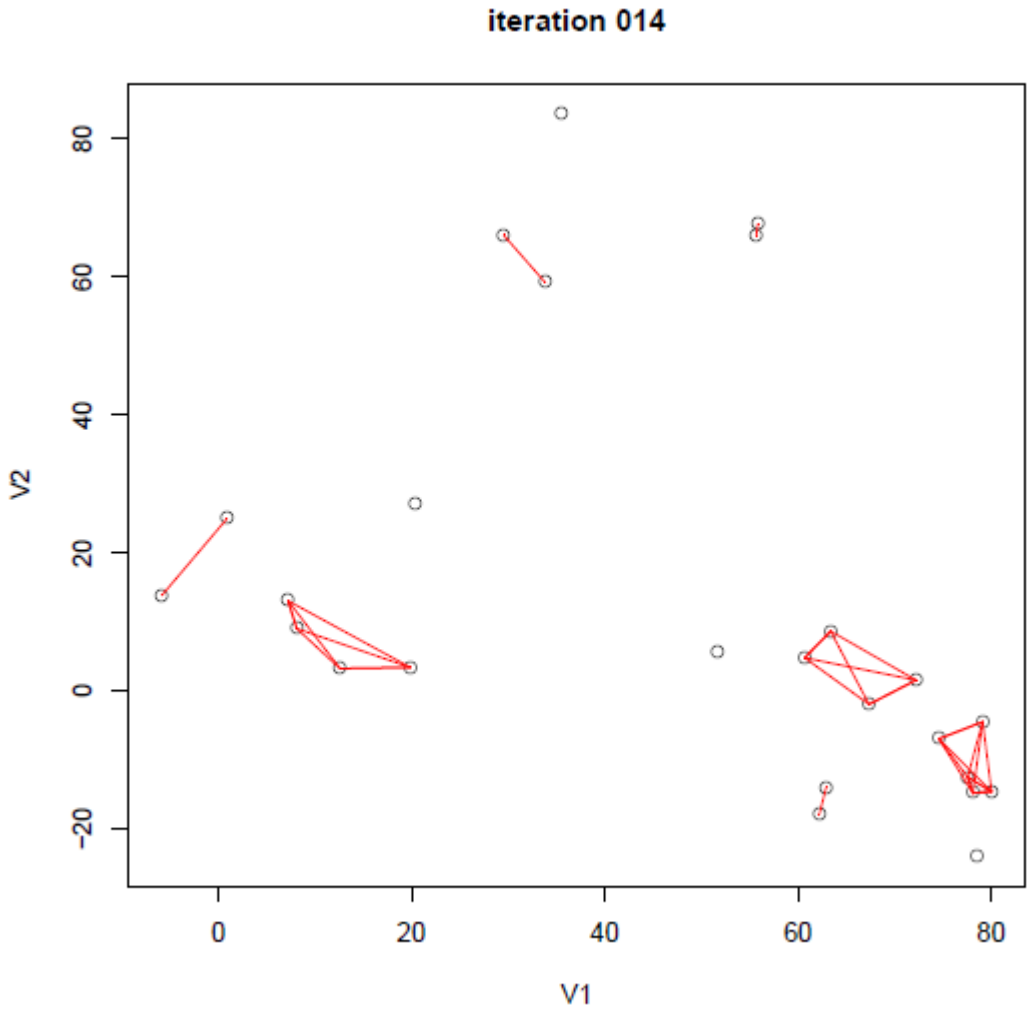
Example



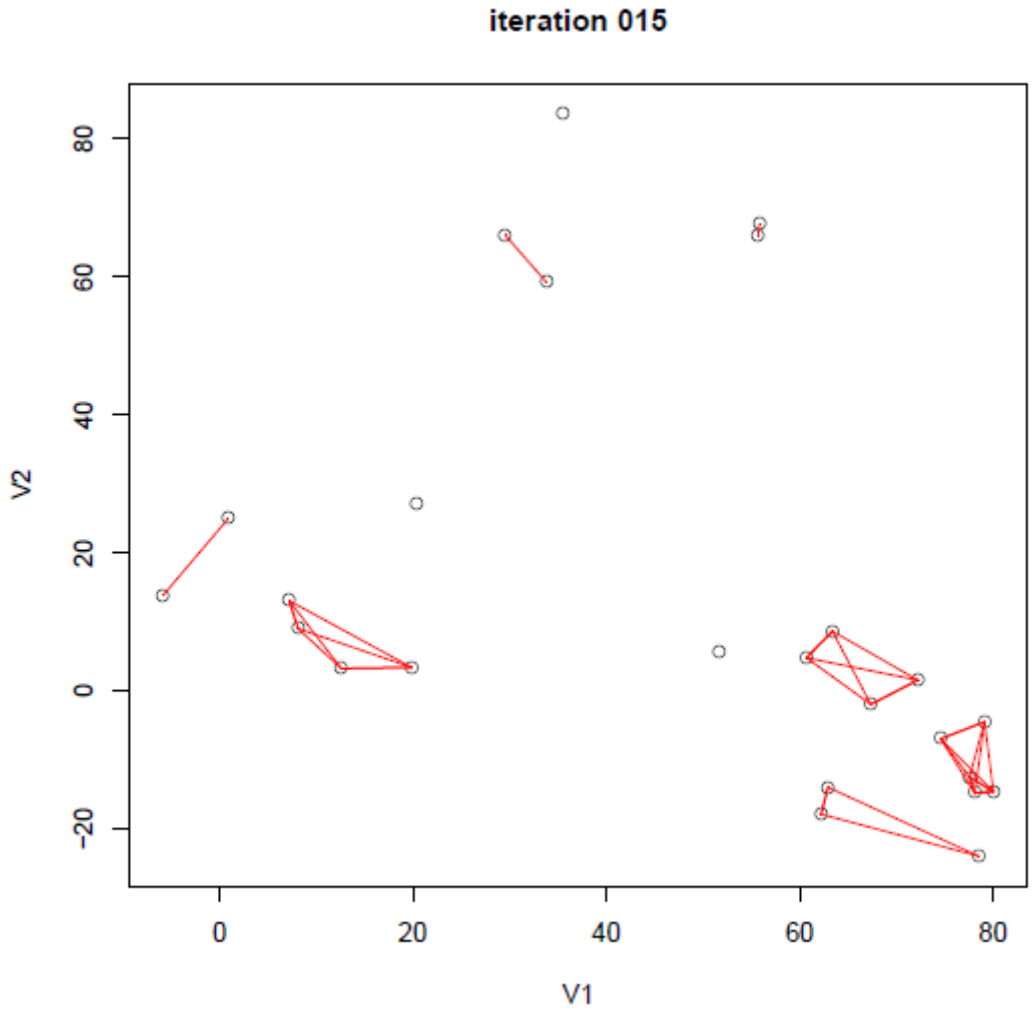
Example



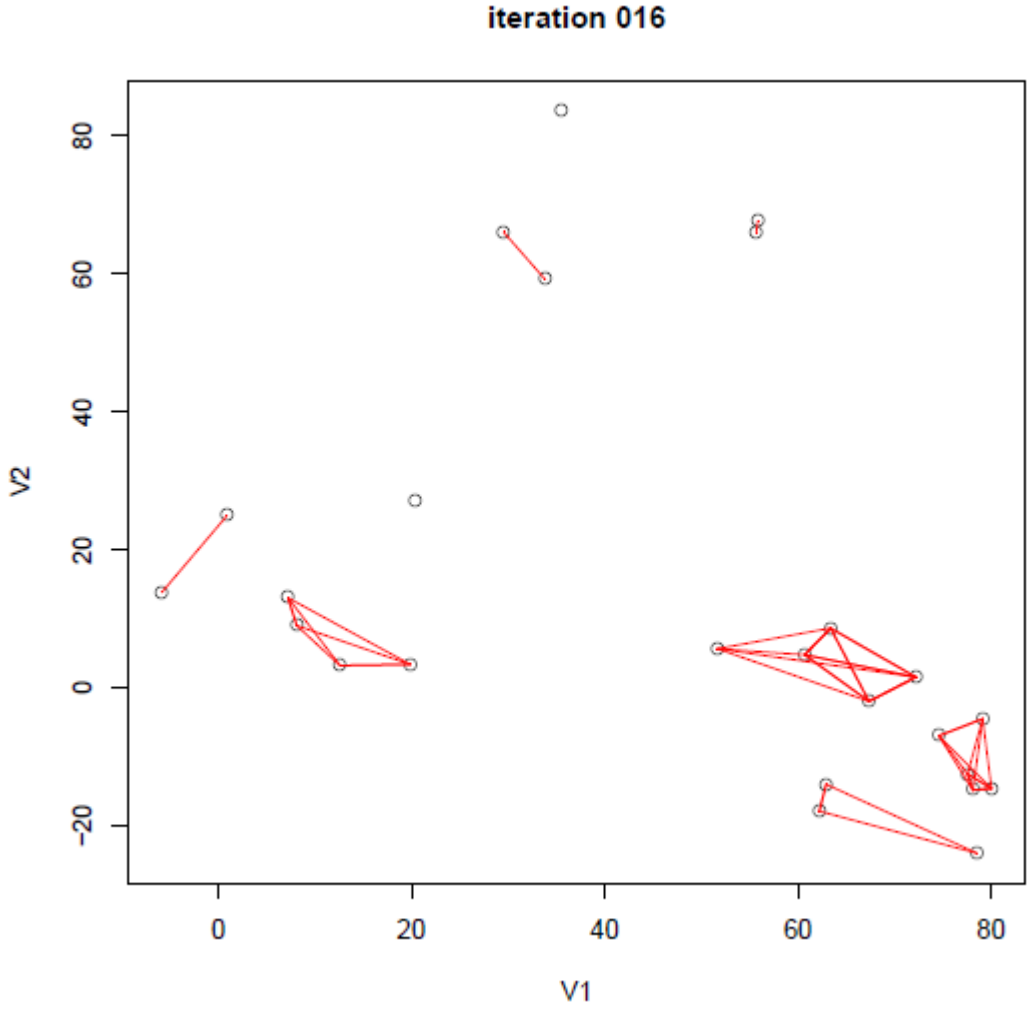
Example



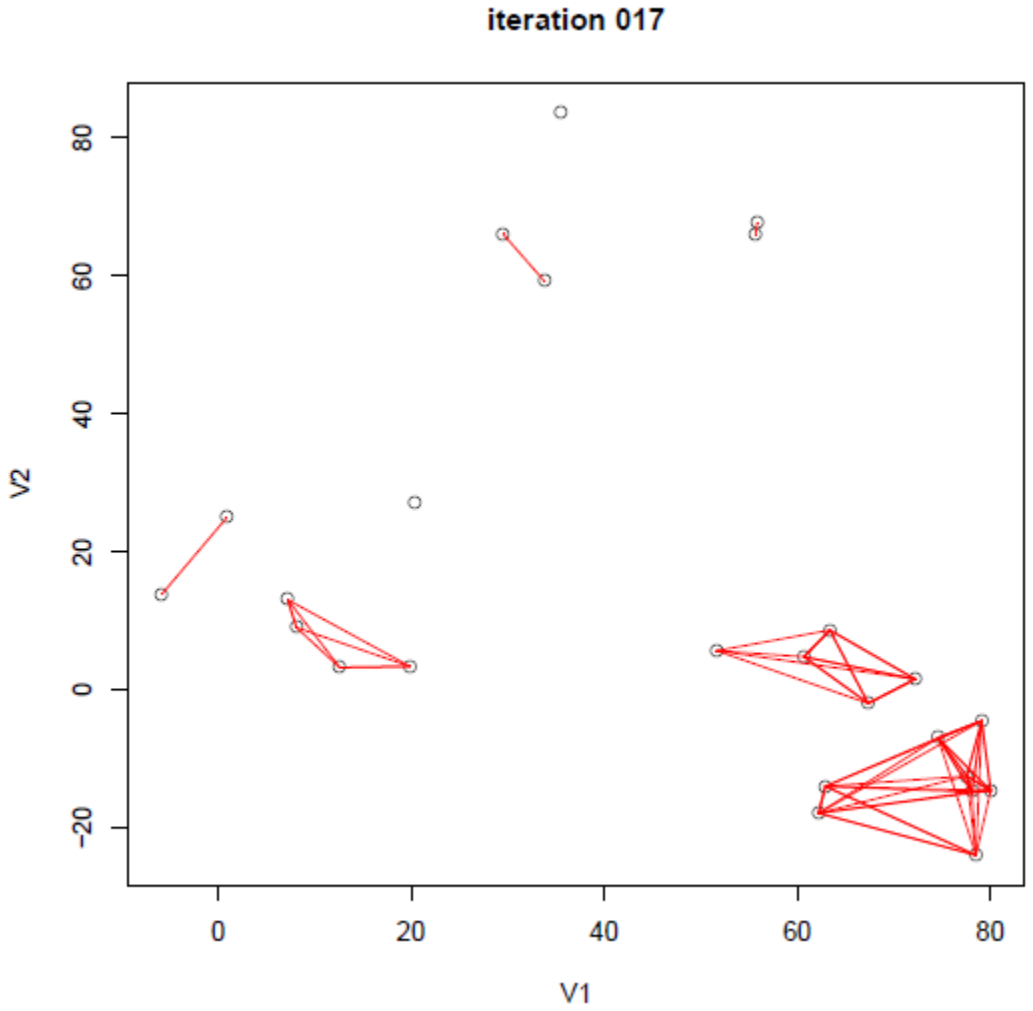
Example



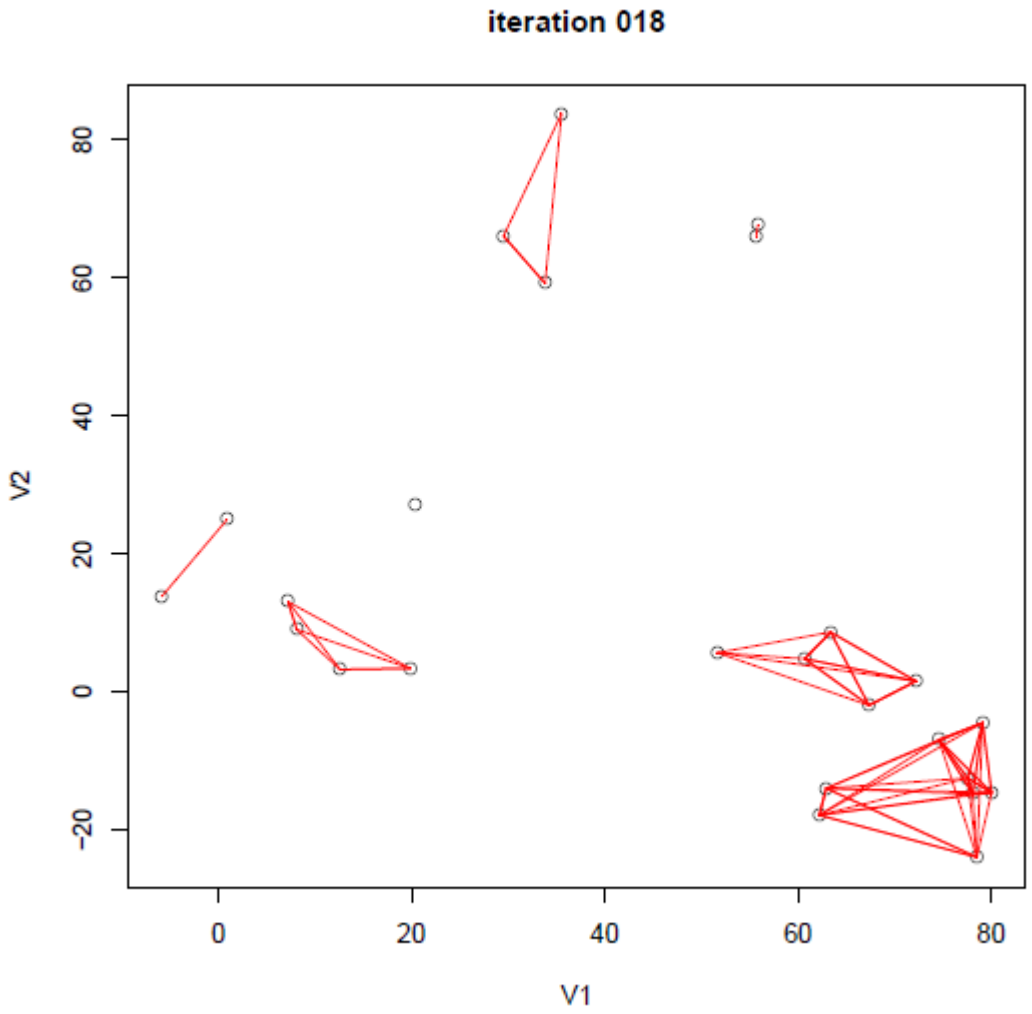
Example



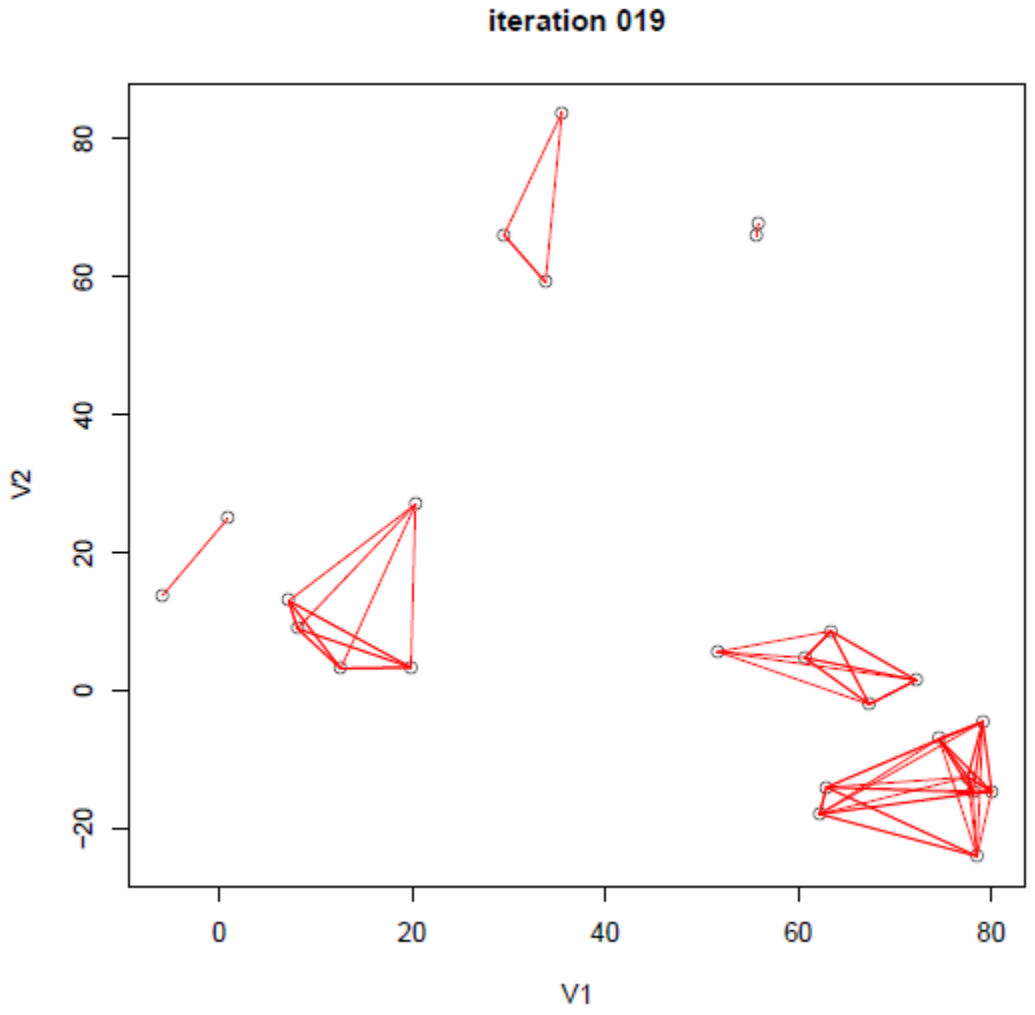
Example



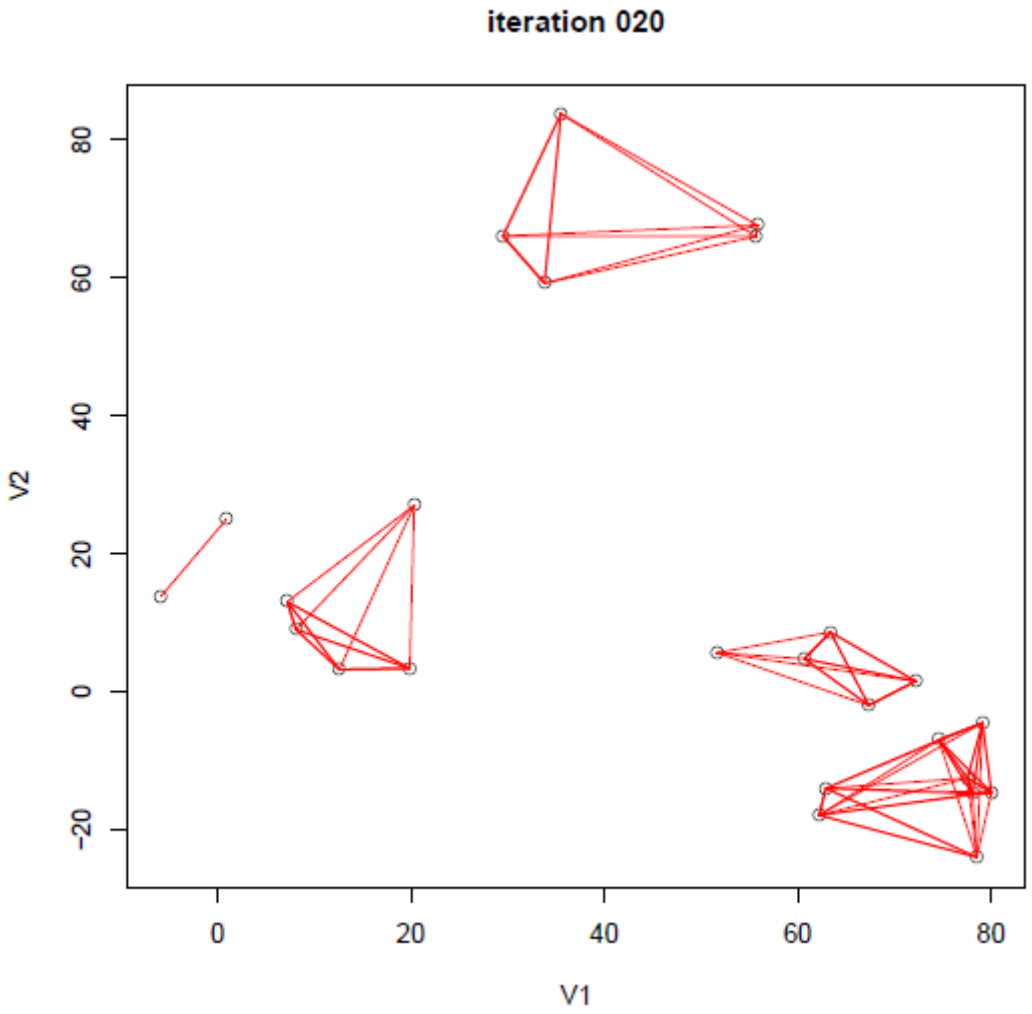
Example



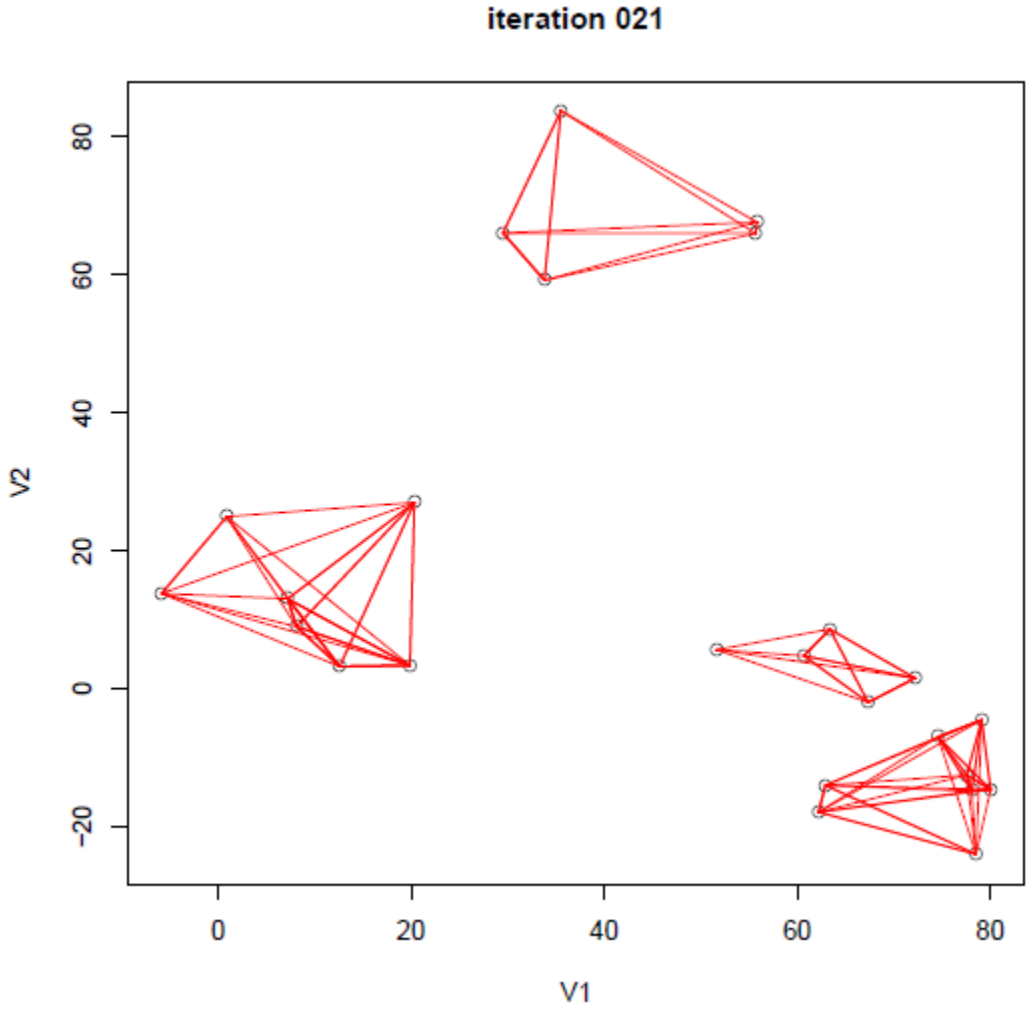
Example



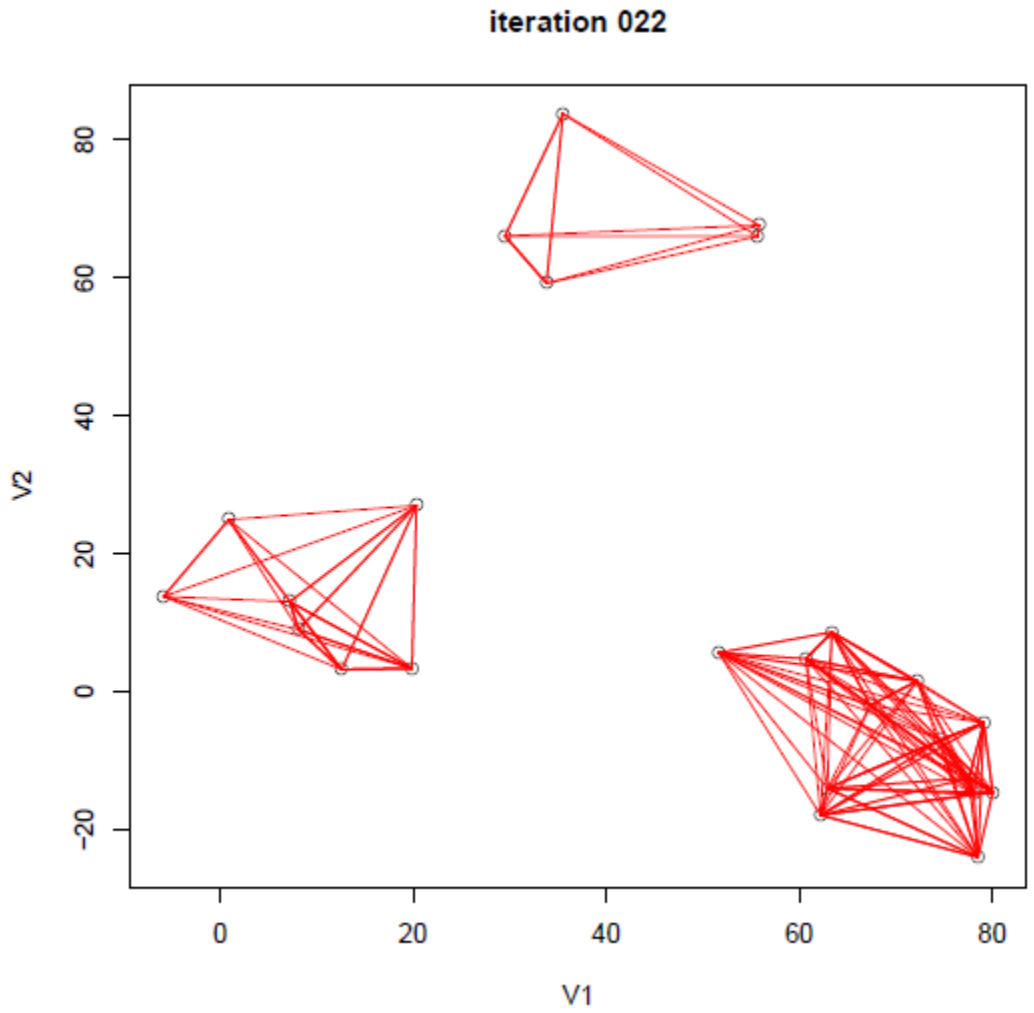
Example



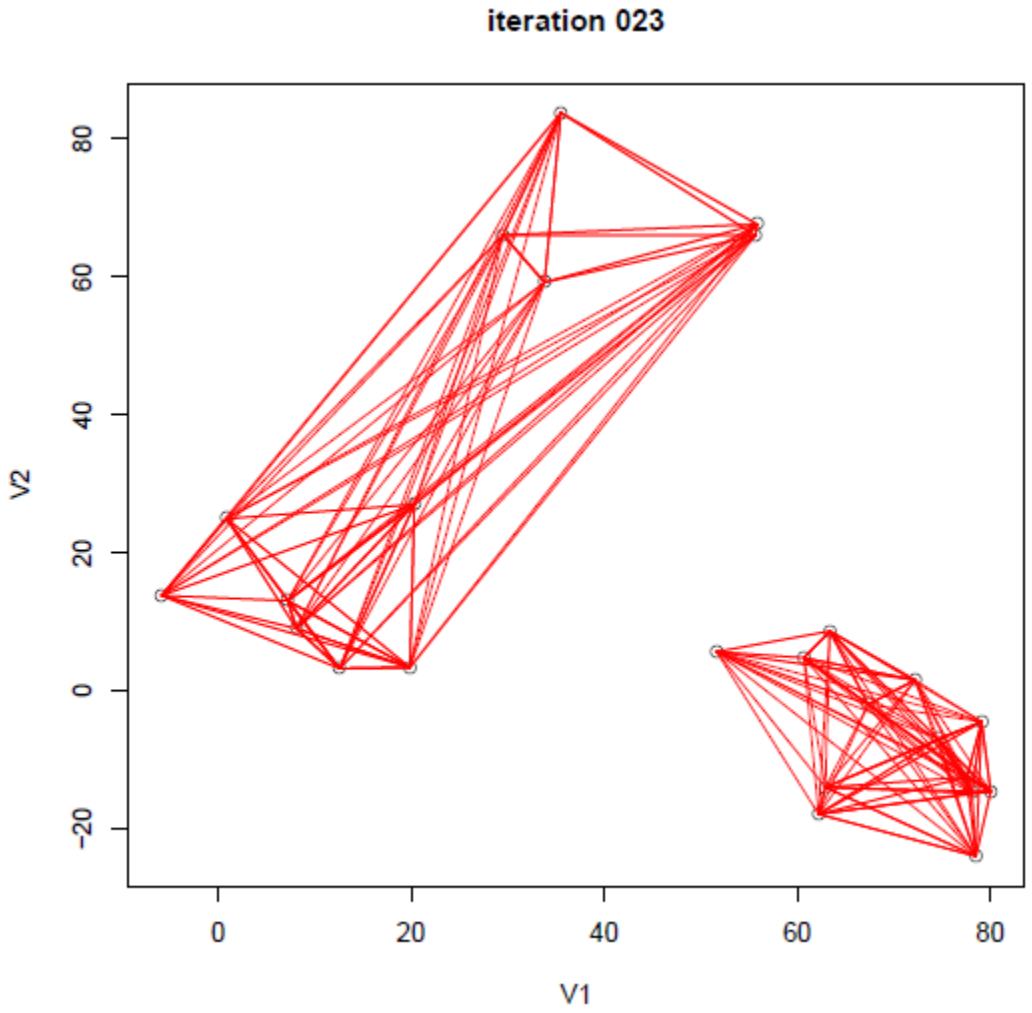
Example



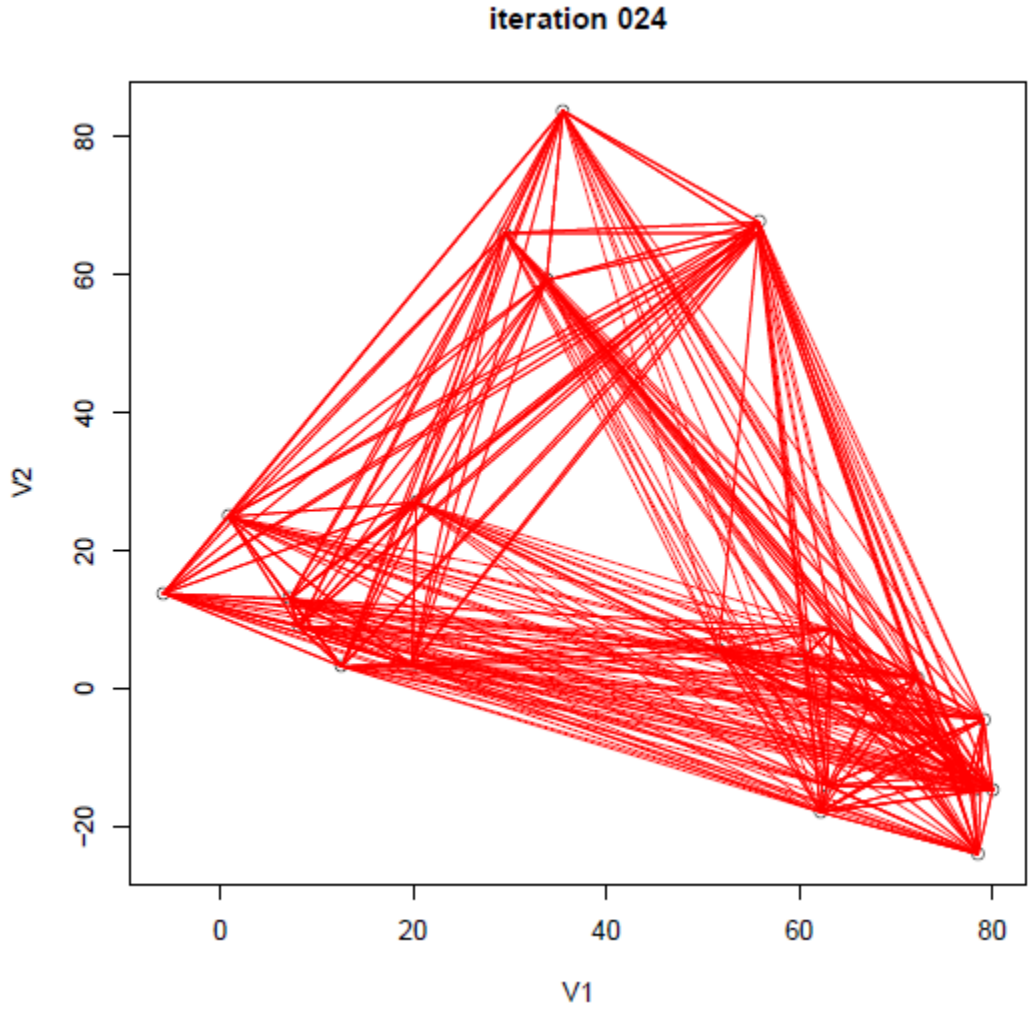
Example



Example



Example



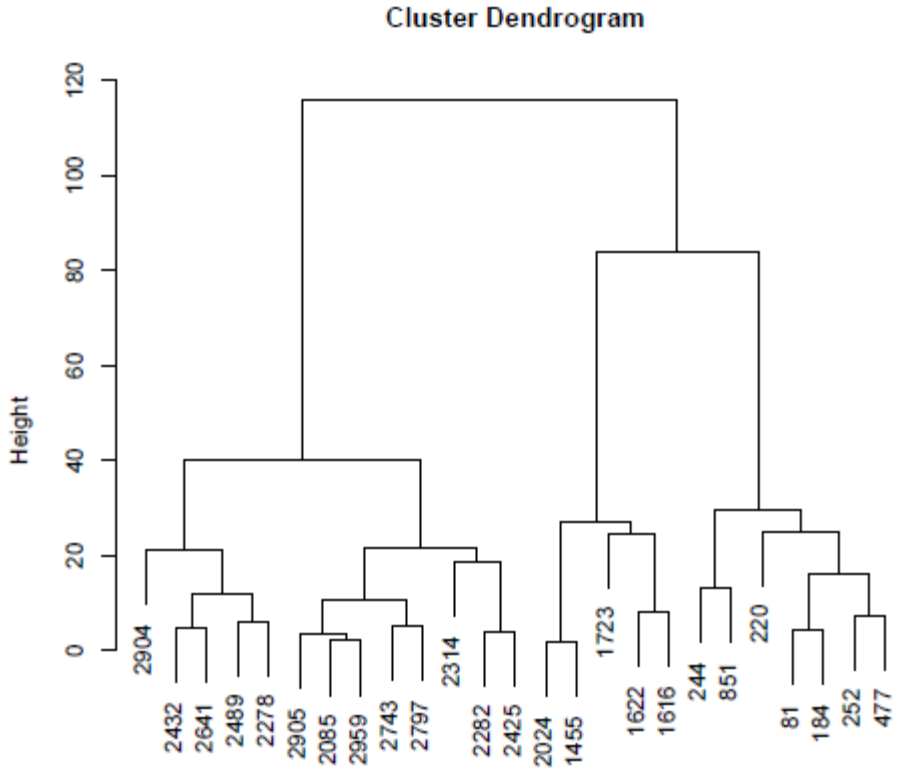
Agglomerative clustering

- Each level of the resulting tree is a segmentation of the data
- The algorithm results in a *sequence* of groupings
- It is up to the user to choose a ”natural” clustering from this sequence

Dendrogram

- Agglomerative clustering is monotonic
  - The similarity between merged clusters is monotone decreasing with the level of the merge
- *Dendrogram*: Plot each merge at the similarity between the two merged groups
- Provides an interpretable visualization of the algorithm and data
- Useful summarization tool, part of why hierarchical clustering is popular

Dendrogram of example data



Groups that merge at high values relative to the merger values of their subgroups are candidates for natural clusters.

Group Similarity

- Given a distance measure between points, the user has many choices for how to define intergroup similarity.
- Three most popular choices
  - Single-linkage: the similarity of the closest pair

$$dist(K_i, K_j) = \min dist(t_{ip}, t_{jq})$$

  - Complete linkage: the similarity of the furthest pair

$$dist(K_i, K_j) = \max dist(t_{ip}, t_{jq})$$

  - Group average: the average similarity between groups

$$dist(K_i, K_j) = avg\ dist(t_{ip}, t_{jq})$$

- Other choices
  - Centroid
  - Medoid

Properties of intergroup similarity

- Single linkage can produce “chaining,” where a sequence of close observations in different groups cause early merges of those groups

- Complete linkage has the opposite problem. It might not merge close groups because of outlier members that are far apart.

- Group average represents a natural compromise, but depends on the scale of the similarities. Applying a monotone transformation to the similarities can change the results

Caveats

- Hierarchical clustering should be treated with caution
- Different decisions about group similarities can lead to vastly different dendrograms.
- The algorithm imposes a hierarchical structure on the data, even data for which such structure is not appropriate.

DBSCAN

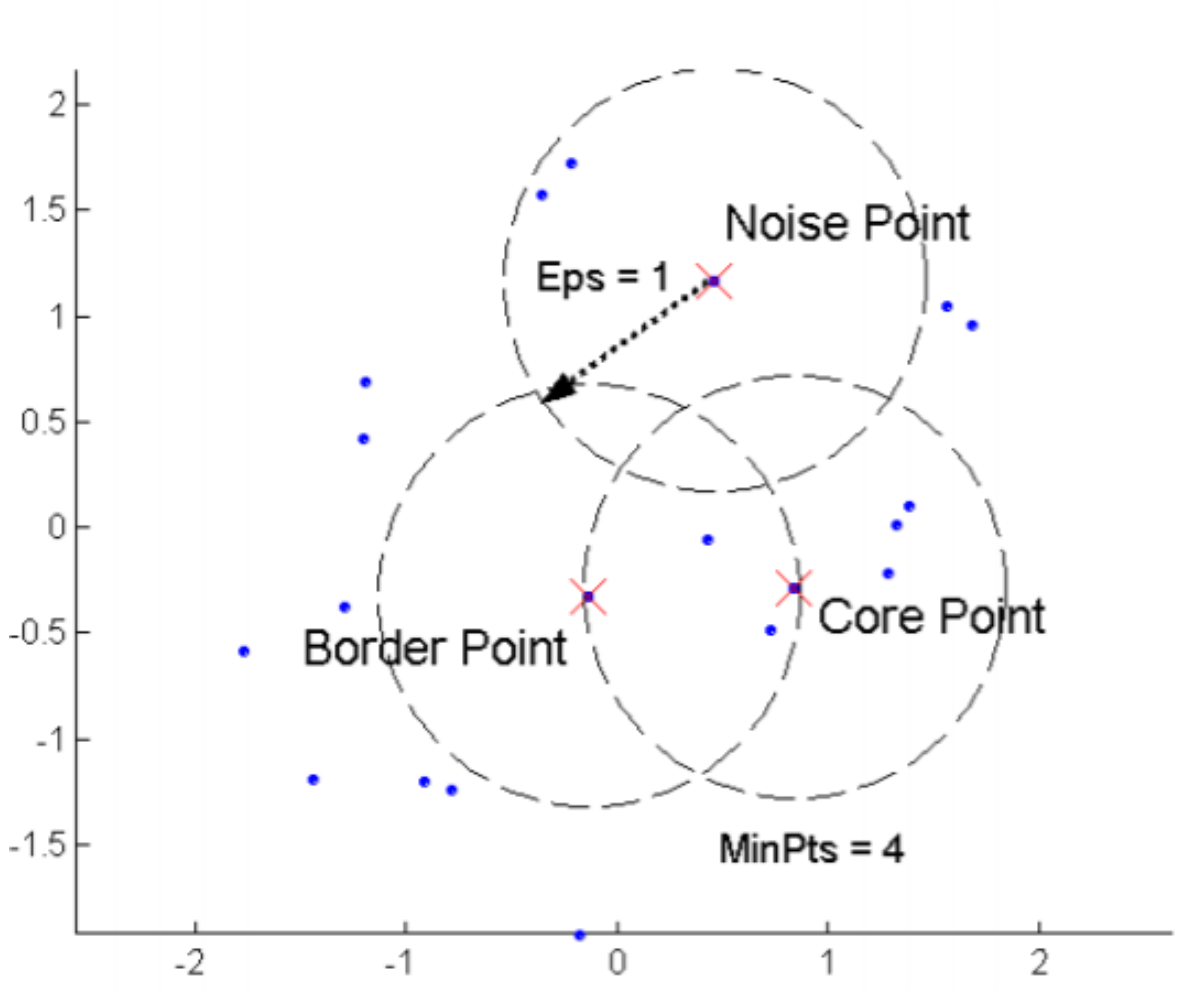
Density-Based Clustering

- Form clusters based on density (local cluster criterion), such as density-connected points
- Each cluster has a considerable higher density of points than outside of the cluster

DBSCAN

- DBSCAN is a density-based algorithm
- Density = number of points within a specified radius r (Eps)
- A point is a *core* point if it has more than a specified number of points (MinPts) within Eps
  - These are points that are at the interior of a cluster
- A *border* point has fewer than MinPts within Eps, but is in the neighborhood of a core point
- A *noise* point is any point that is not a core point or a border point

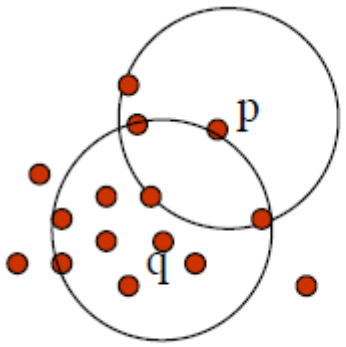
DBSCAN: Core, Border, and Noise points



DBSCAN

- Two parameters:
  - *Eps*: Maximum radius of the neighborhood
  - *MinPts*: Minimum number of points in an Eps-neighborhood of that point
- $N_{Eps}(q)$: {p belongs to D | dist(p,q) ≤ Eps}
- Directly density-reachable: A point *p* is directly density-reachable from a point *q* w.r.t. Eps, MinPts if
  - *p* belongs to $N_{Eps}(q)$
  - *q* is a core point, core point condition:

$$|N_{Eps}(q)| \geq MinPts$$

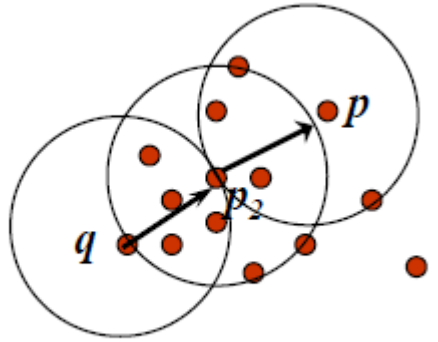


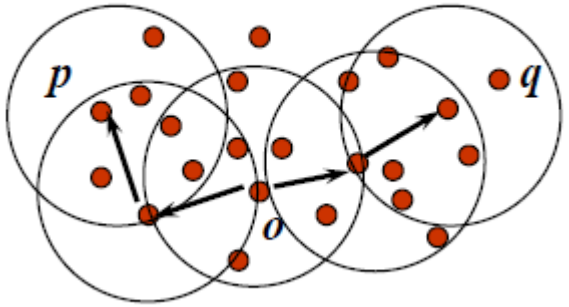
MinPts = 5

Eps = 1 cm

Density-Reachable and Density-Connected

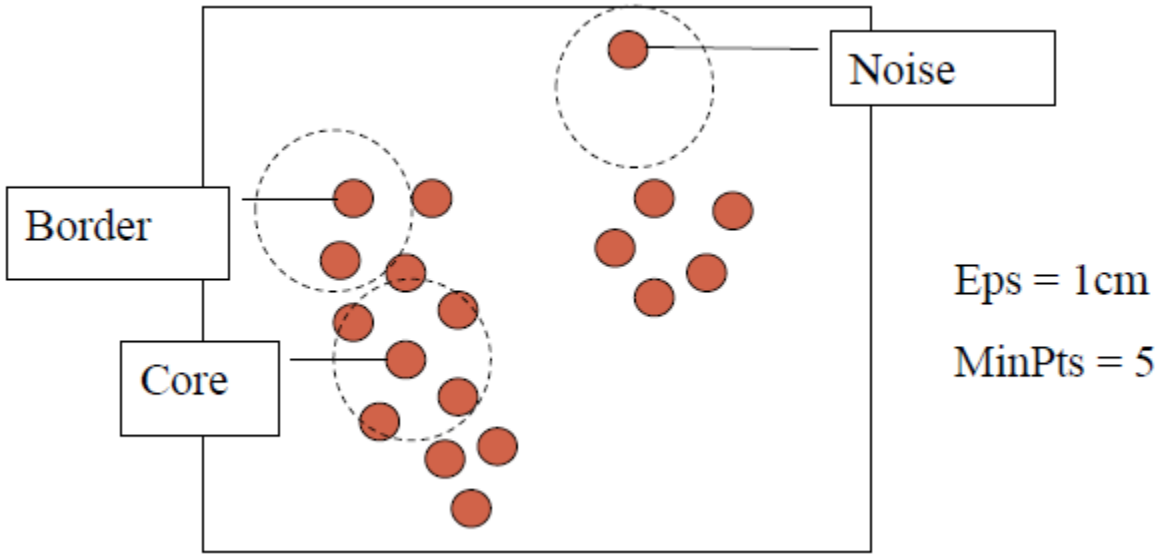
- Density-reachable:
  - A point *p* is density-reachable from a point *q* w.r.t. *Eps, MinPts* if there is a chain of points $p_1, \ldots, p_n$; $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i
- Density-connected
  - A point *p* is density-connected to a point *q* w.r.t. Eps, MinPts if there is a point *o* such that both *p* and *q* are density-reachable from *o* w.r.t. Eps and MinPts





DBSCAN

- Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Noise: object not contained in any cluster is noise
- Discovers clusters of arbitrary shape in spatial databases with



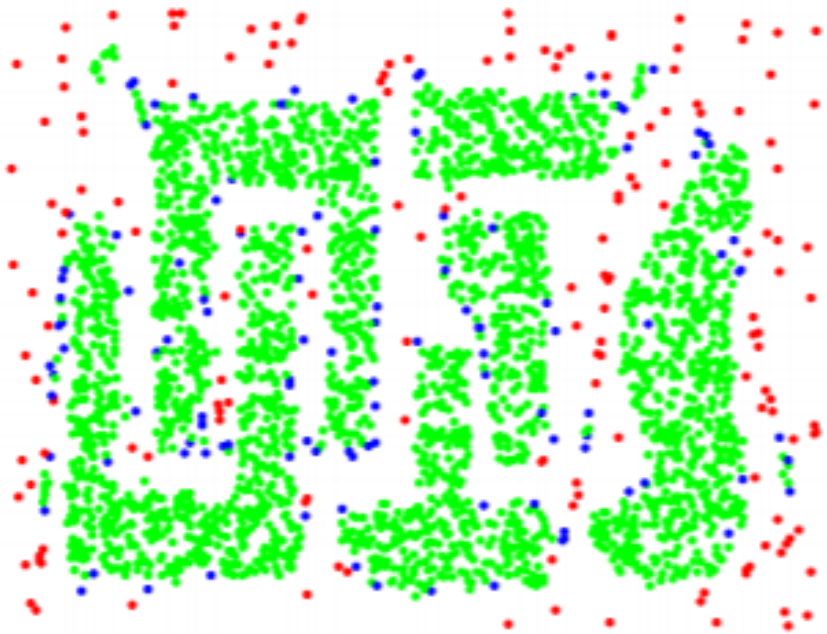
DBSCAN: The algorithm

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the ε-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the ε-neighborhood of p;
(8)          for each point p' in N
(9)              if p' is unvisited
(10)                 mark p' as visited;
(11)                 if the ε-neighborhood of p' has at least MinPts points,
                     add those points to N;
(12)             if p' is not yet a member of any cluster, add p' to C;
(13)         end for
(14)         output C;
(15)     else mark p as noise;
(16) until no object is unvisited;
```

- If a spatial index is used, the computational complexity of DBSCAN is O(nlogn), where n is the number of database objects. Otherwise, the complexity is $O(n^2)$

DBSCAN: Large Eps

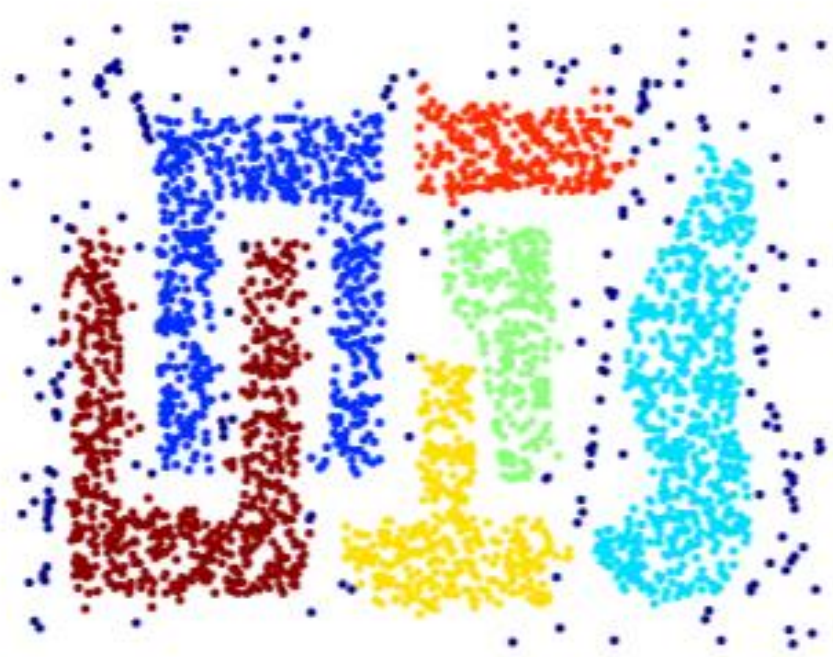
Original Points



Point types: core (green), border (blue), and noise (red)

DBSCAN: Optimal Eps

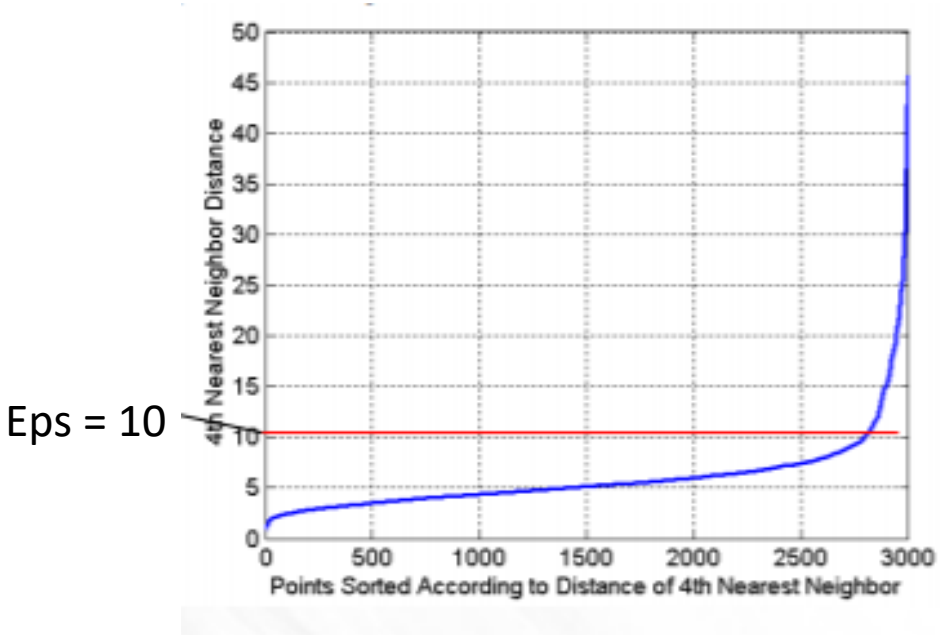
Original Points



Clusters

Determining Eps and MinPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- Can plot sorted distance of every point to its k^{th} nearest neighbor



DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

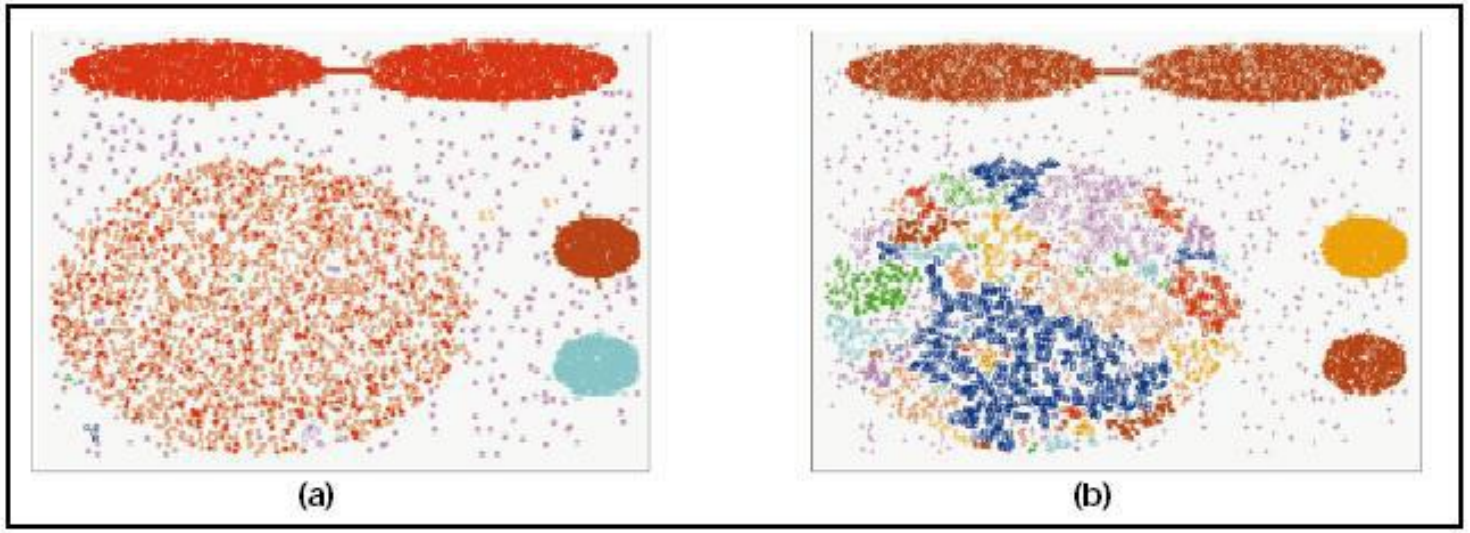
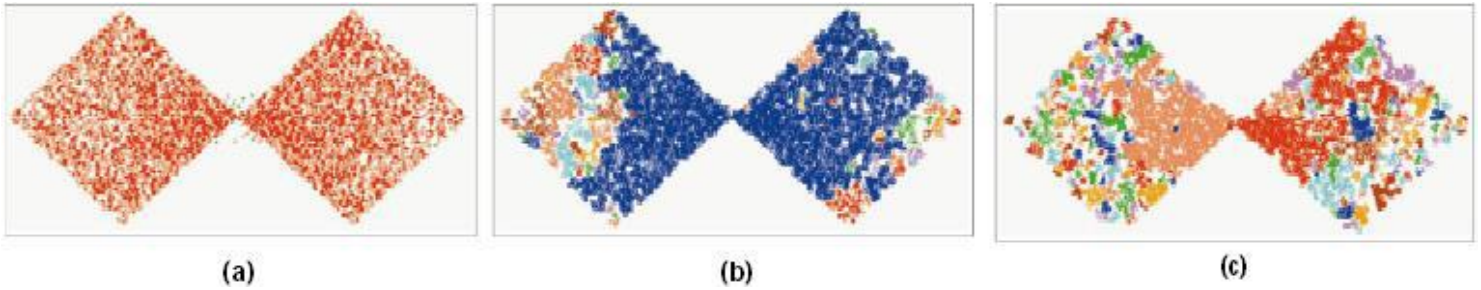


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



Applications

Applications of Clustering

- Things in a cluster share similar properties
  - Contents of data
  - Communities in social networks (Social Network Analysis)
  - Relevance of documents (Information Retrieval)
  - Meanings of photos (Multimedia)
  - Locations of geo-tagged objects
  - Styles of music
- It implies
  - (Properties Discovery)
    A cluster might represent some specific properties
  - (Properties Inference)
    We can infer properties of unknown data from same-cluster data.

Properties Discovery – POI Identification

- Identify point-of-interests (special locations) to geo-located photos.
- Photos in close locations (same cluster) represent the same POI.

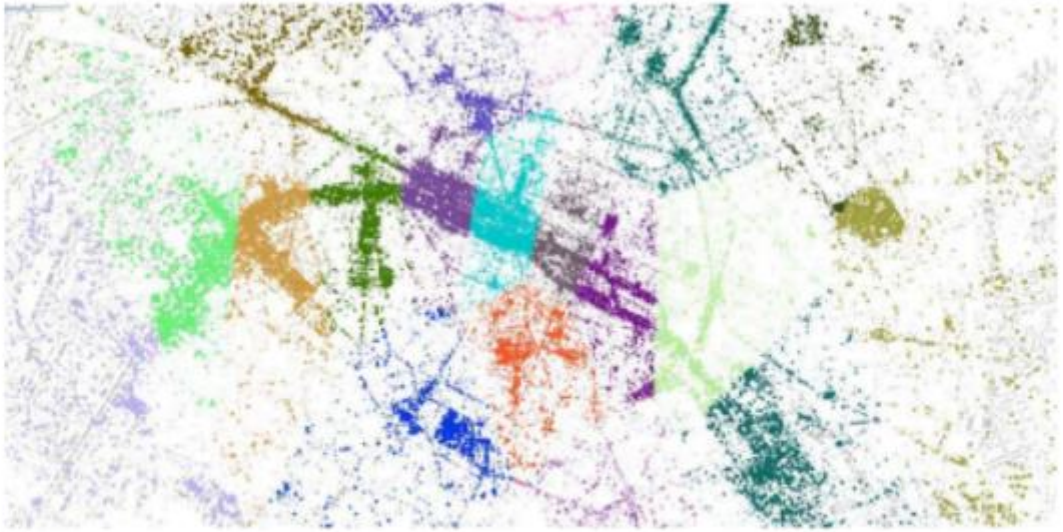


Figure : Identify POIs to Photos with Clustering [Yang et al., SIGIR 2011]

Properties Discovery – Community Detection

- Identify nodes in social networks into several communities.
- Nodes in a community have more interactions and connections.
- Cluster nodes into groups with related information.
- Nodes in a cluster represent a community.

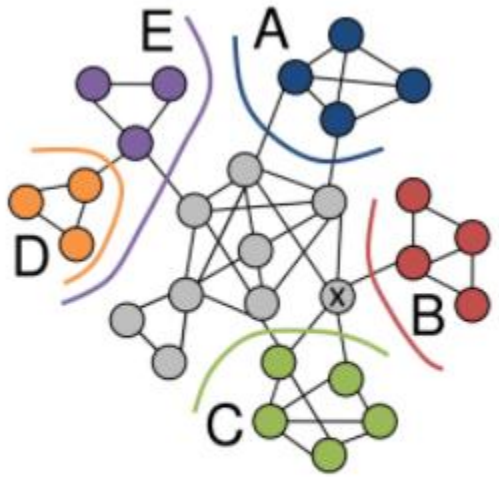


Figure : Detect communities by clustering [Leskovec et al., WWW 2010]

Properties Discovery – Type Discovery

- While a document describes an entity, it might have types.
- Same-cluster documents might share same types

Born	Thomas Cruise Mapother IV July 3, 1962 (age 51) Syracuse, New York, United States
Occupation	Actor, producer, writer
Years active	1981–present
Religion	Scientology
Spouse(s)	Mimi Rogers (m. 1987–90) Nicole Kidman (m. 1990–2001) Katie Holmes (m. 2006–12)
Children	3 (two adopted)
Website	
	TomCruise.com

Figure : Discover types of Wikipages by clustering infoboxes [Nguyen et al., CIKM '12]

Clustering in Information Retrieval

- Core Problem of Information Retrieval
  - Given many documents and a query, rank them by their relevance.
- Clustering Hypothesis
  - Closely associated documents tend to be relevant to the same requests

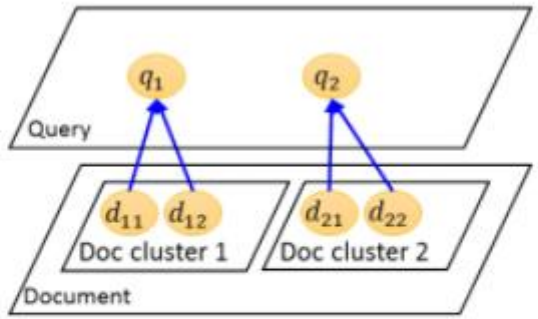


Figure : Illustration of clustering hypothesis.

Search Results Clustering

- Cluster search results into several groups.
- Relevance of documents can inference to same-cluster documents.
- More effective information presentation to users.

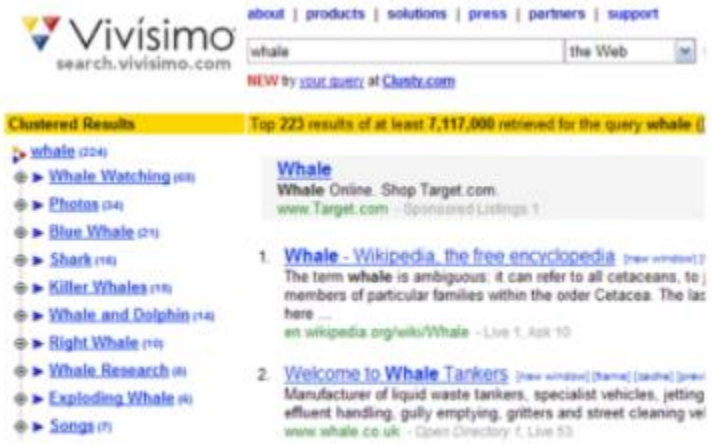
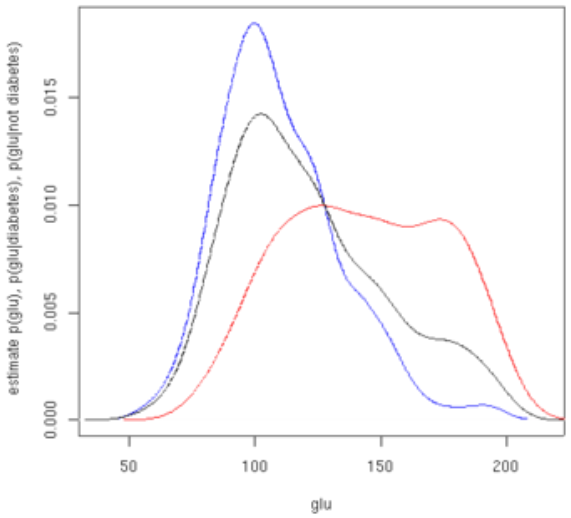


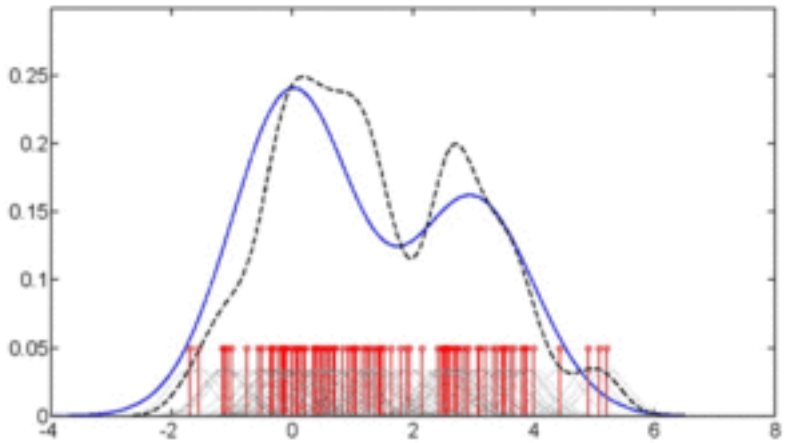
Figure : Vivisimo, a search engine clustering search results.

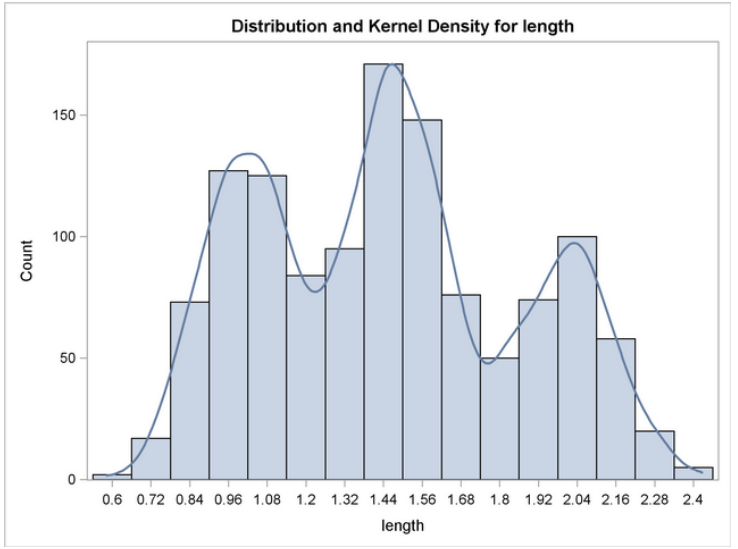
Density Estimation

Review of Density Estimation

- Goal
  - Estimate the density function for a random variable from data
- Can be considered as an extension of histogram

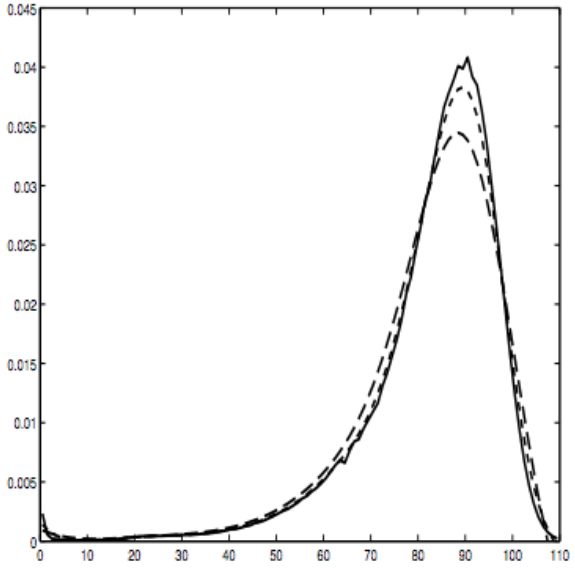




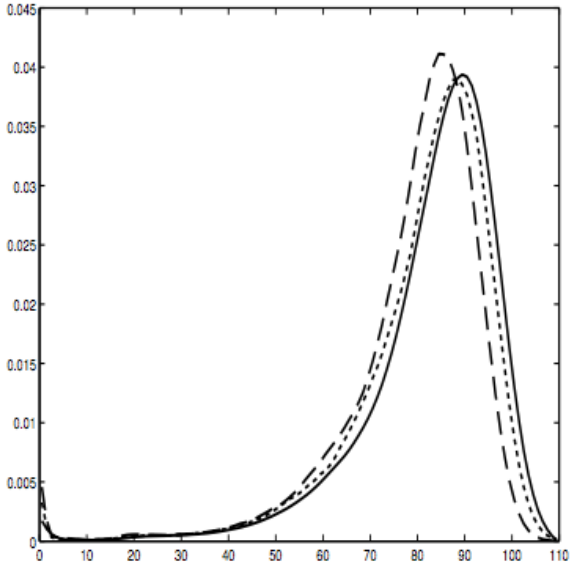


Example: Mortality Estimation

- Country-specific period lifetables can be utilized.
- Different bandwidths decide the smoothness
- Can detect distribution shifting



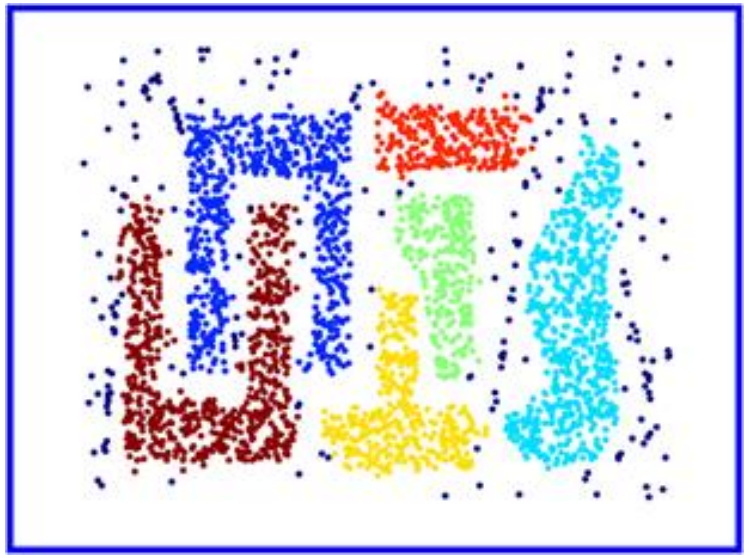
(a) The solid, short-dashed and long-dashed lines correspond to smoothed histograms using bandwidths of 0.5, 2 and 3, respectively.

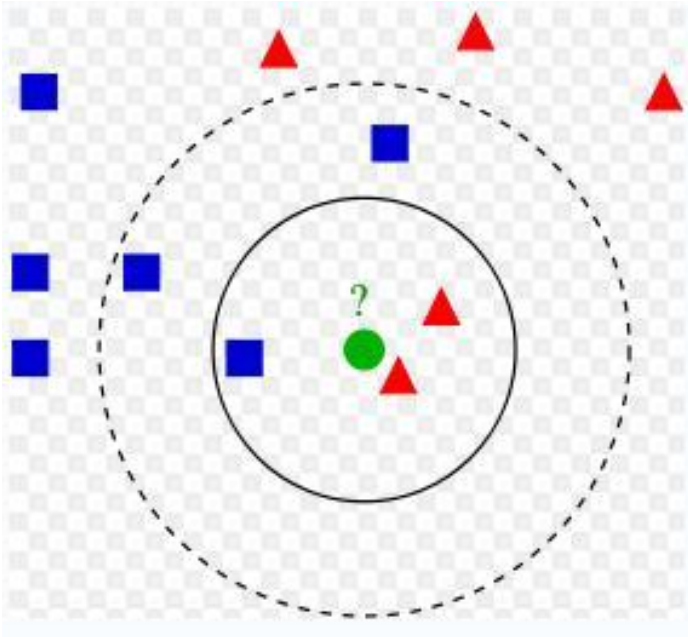


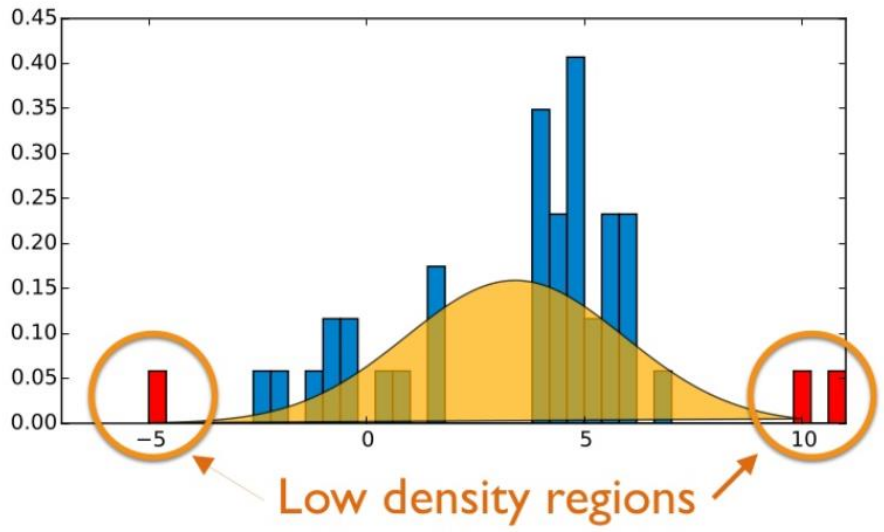
(b) 1990 (long-dash), 2000 (dotted) and 2010 (solid)

Many Applications

- Density-based clustering
  - E.g., DBSCAN
- Classification
  - E.g., KNN, what's the best K?
- Outlier Detection

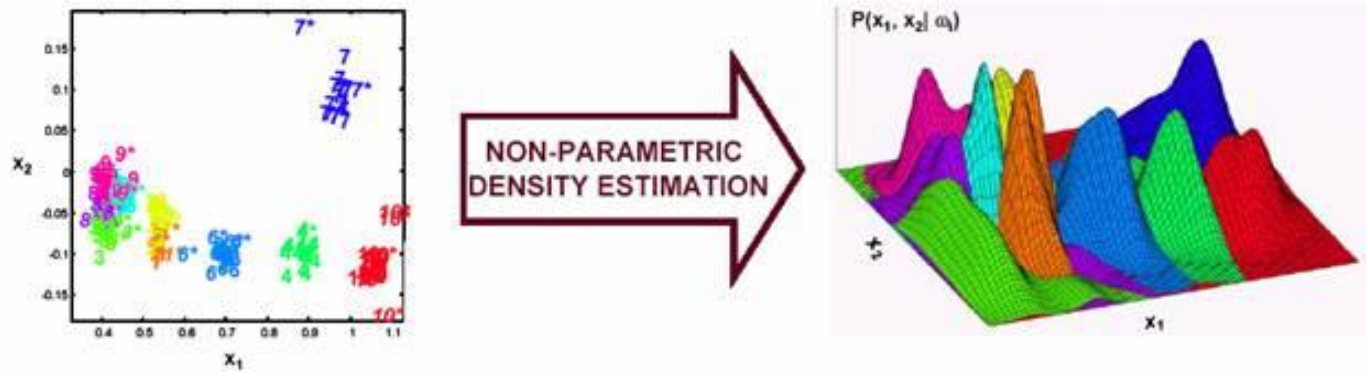




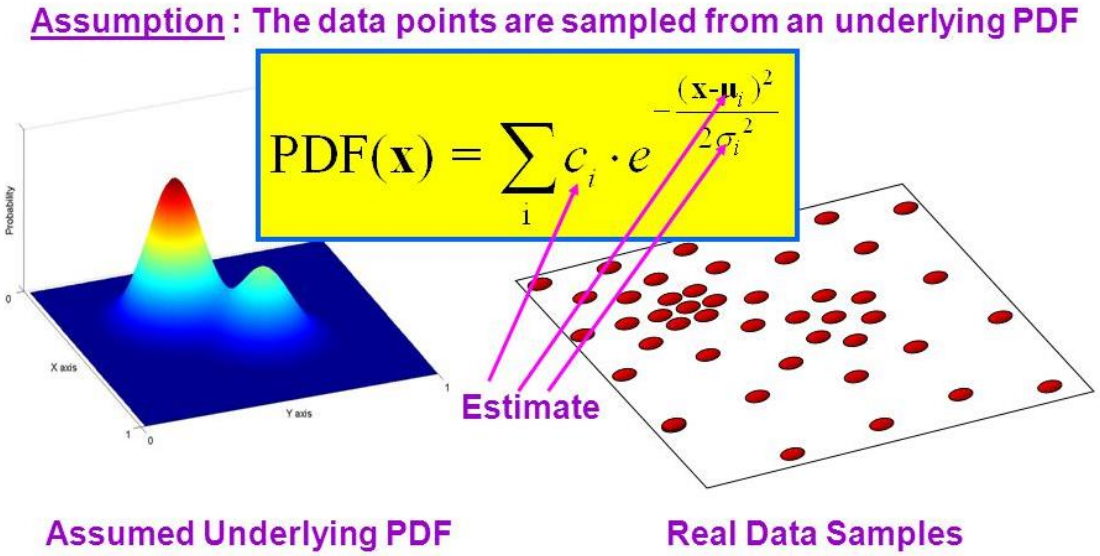


Nonparametric and parametric methods

- Nonparametric methods
  - No assumption about the forms of the underlying densities
  - Applicable to any distribution

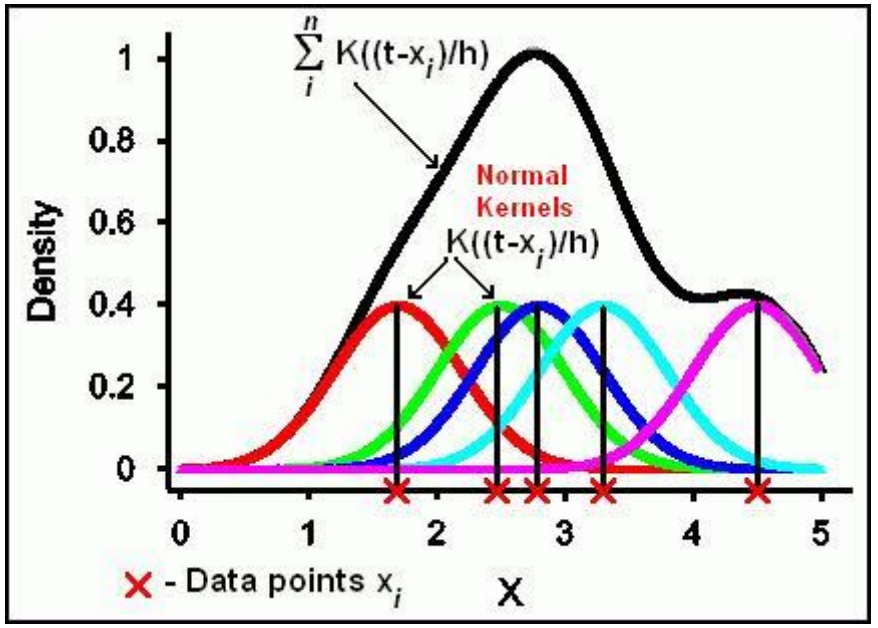


- Parametric methods
  - Have assumptions about the forms of the underlying densities
  - Determined by fixed but unknown parameters



Kernel Density Estimation

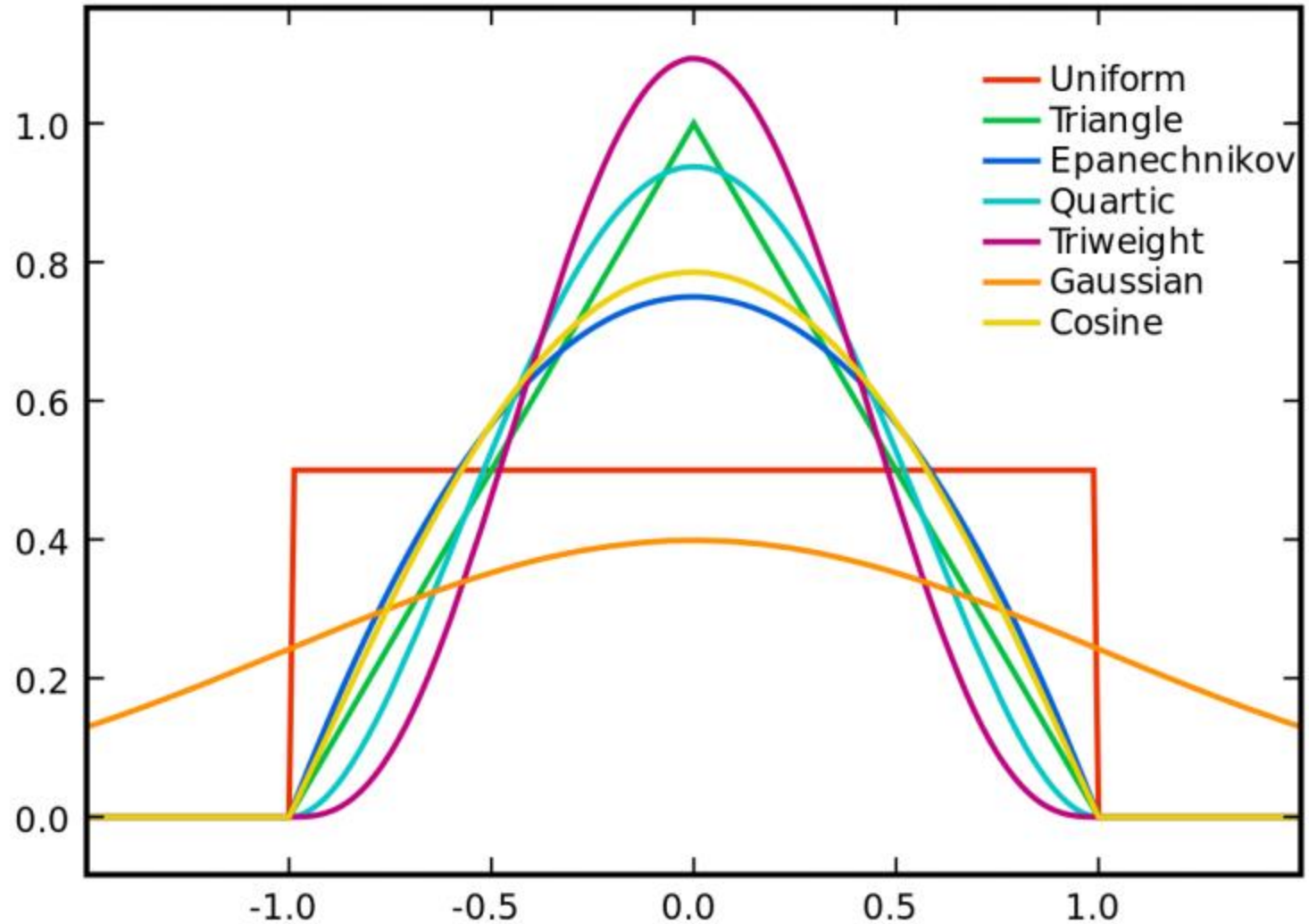
- A non-parametric method for density estimation
- A known density function (kernel) is averaged across the observed data to create a smooth approximation.



Kernel Density Estimation

- Given a dataset $D = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, estimate its density function $f(\boldsymbol{x})$
- Kernel density estimator:
  - $\hat{f}_h(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} K_h(\boldsymbol{x} - \boldsymbol{x}_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h})$
  - h: bandwidth, controlling the smoothness of f
  - K: a non-negative real-valued integrable function, serving as weighting function
    - $\int_{-\infty}^{+\infty} K(u)du = 1\ (normalization)$
    - $K(u) = K(-u)\ for\ all\ u\ (symmetric)$

Examples of Kernels



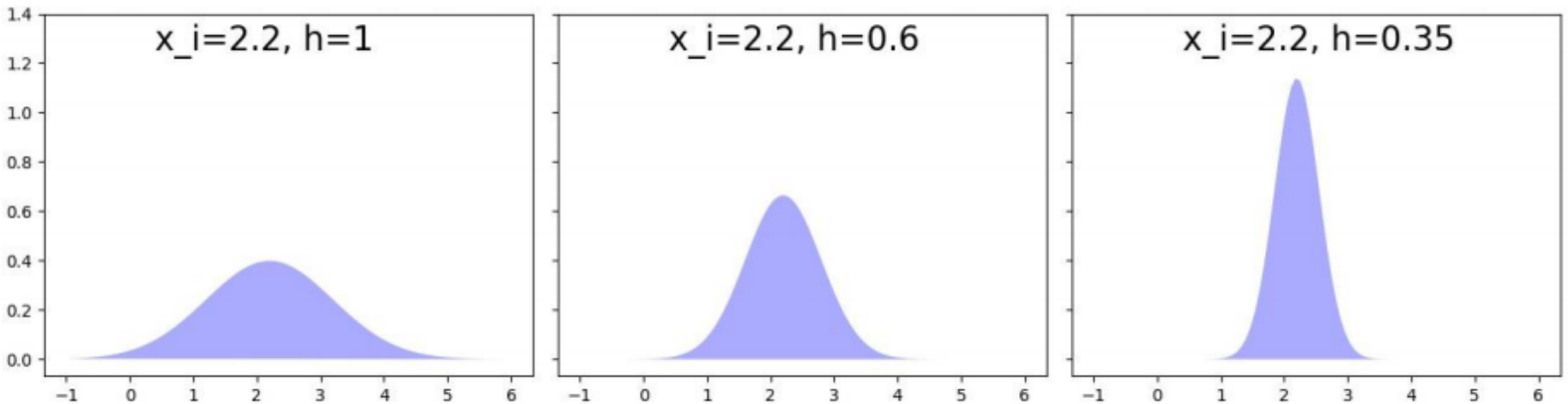
Gaussian Kernel in 1-D case

- Example: Gaussian kernel
  - $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$
- Scaled kernel
  - $K_h(u) = \frac{1}{h} K\left(\frac{u}{h}\right)$
  - In the Gaussian kernel case: $K_h(u) = \frac{1}{h\sqrt{2\pi}} e^{-\frac{u^2}{2h^2}}$

Influence from one data point

- The influence of x_i to x can be considered as a weighting function centered at x_i

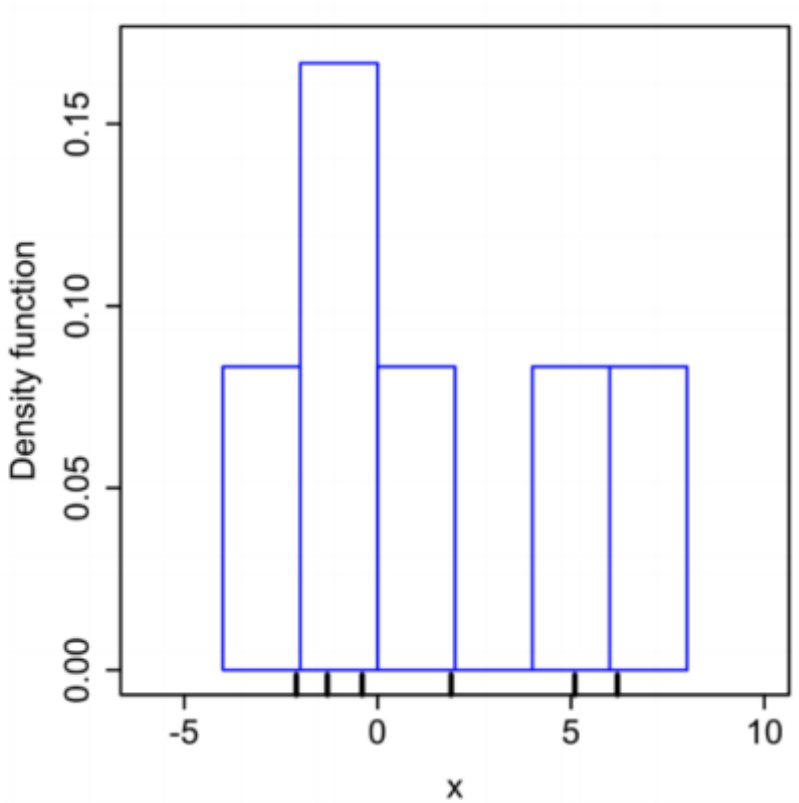
$$K_h(x - x_i) = \frac{1}{h\sqrt{2\pi}} e^{-\frac{(x-x_i)^2}{2h^2}}$$



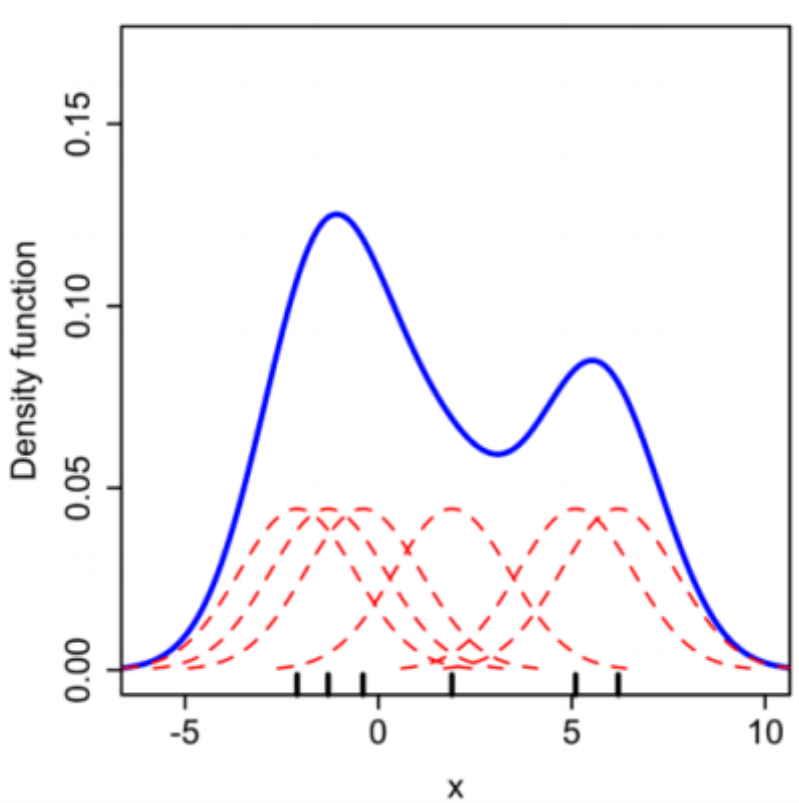
Recall: $\hat{f}_h(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} K_h(\boldsymbol{x} - \boldsymbol{x}_i) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h})$

Influence from multiple data points

- Aggregate influence from multiple data points to x

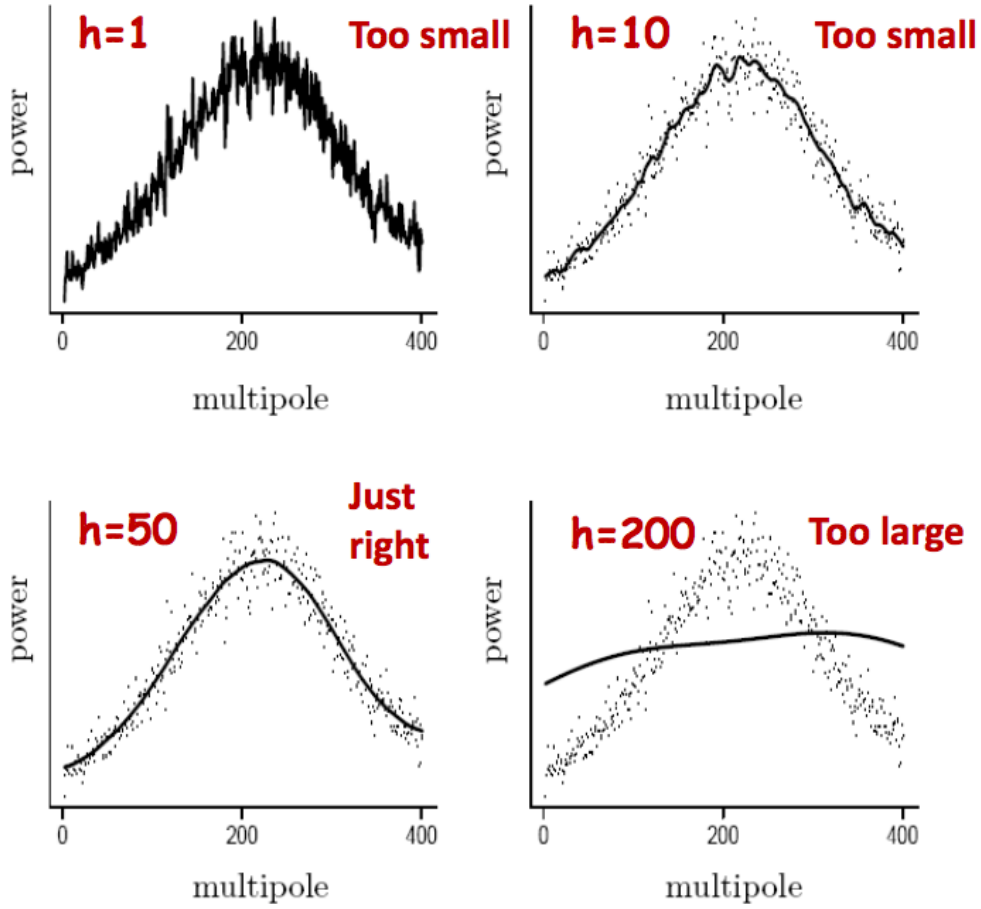


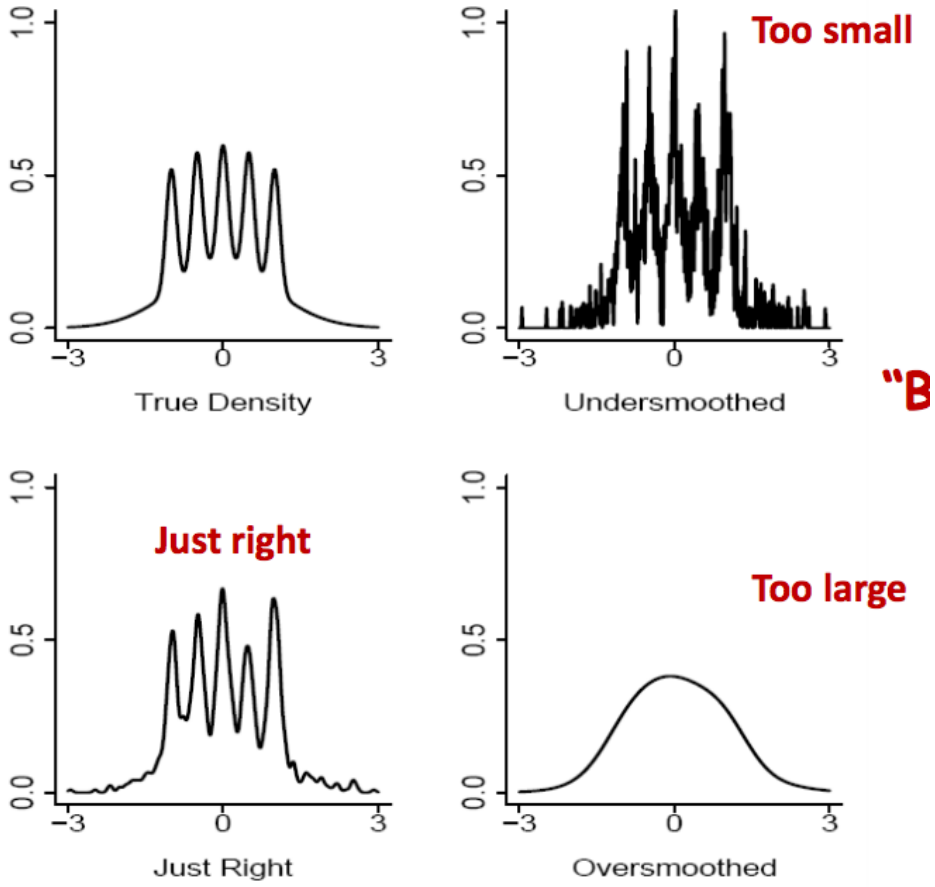
histogram



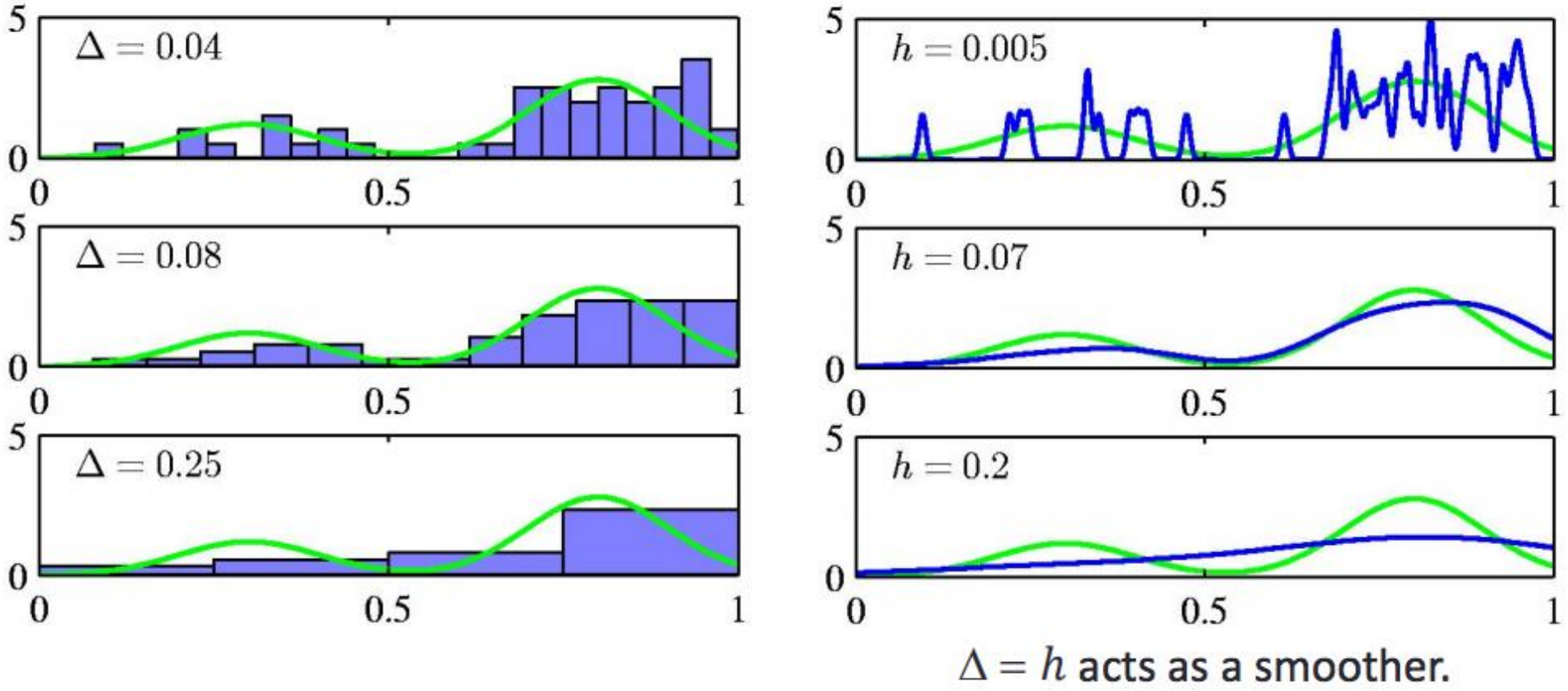
Density: Each red curve indicates $\frac{1}{n}K_h(x - x_i)$

Choice of bandwidth





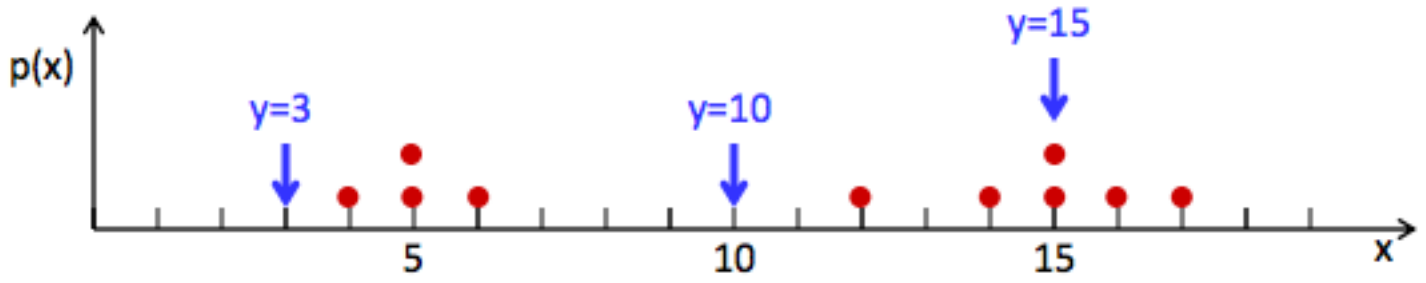
Histograms vs. Kernel Density Estimation



$\Delta = h$ acts as a smoother.

Example

- Given a dataset X = {4, 5, 5, 6, 12, 14, 15, 15, 16, 17}, use Parzen windows to estimate the density $p(x)$ at y = 3,10,15; use h = 4



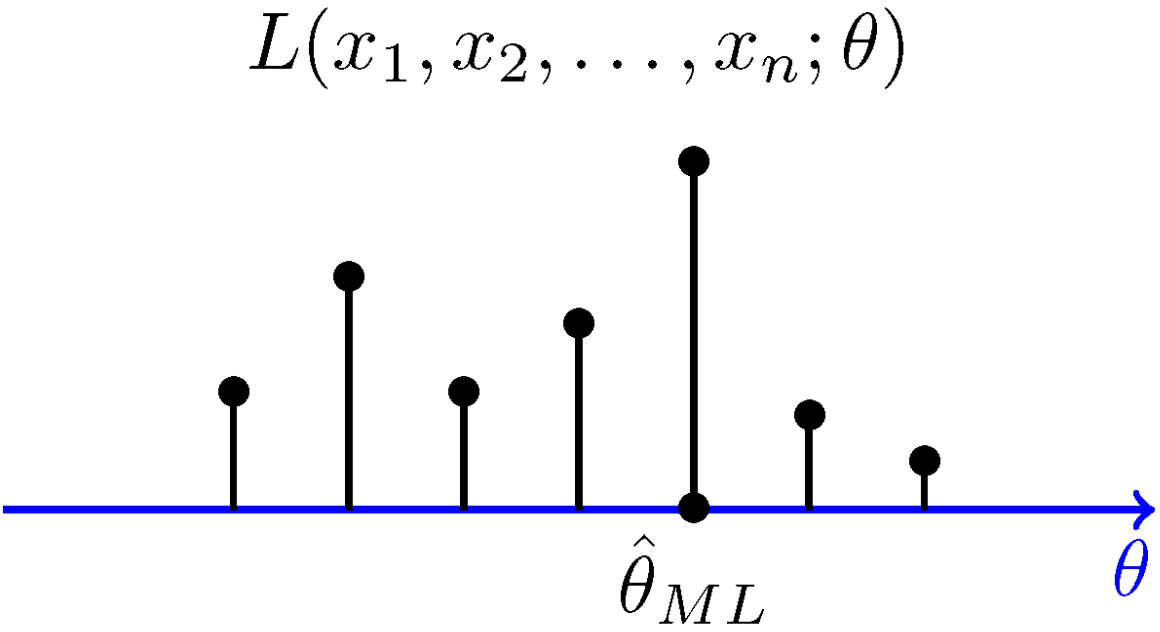
$$p(y=3)=\frac{1}{Nh^D}\sum_{n=1}^{N}K\left(\frac{x-x^{(n}}{h}\right)=\frac{1}{10\times 4^1}\left[K\left(\frac{3-4}{4}\right)+K\left(\frac{3-5}{4}\right)+\cdots K\left(\frac{3-17}{4}\right)\right]=0.0025$$

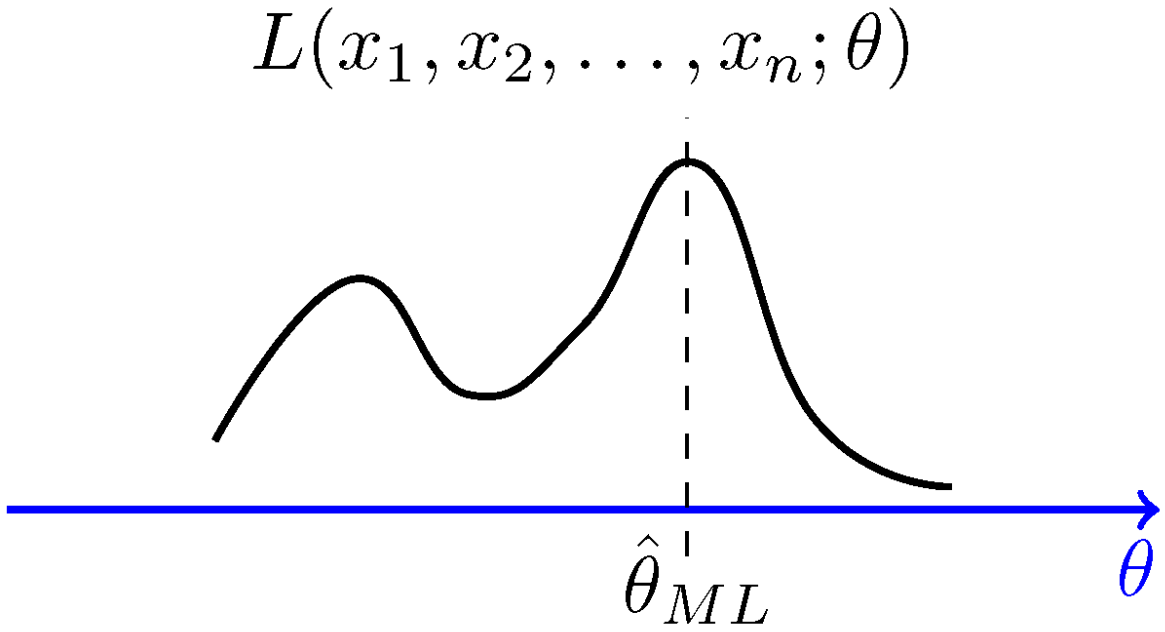
$$p(y=10)=\frac{1}{10\times 4^1}[0+0+0+0+0+0+0+0+0+0]=0$$

$$p(y=15)=\frac{1}{10\times 4^1}[0+0+0+0+0+1+1+1+1+0]=0.1$$

Maximum Likelihood Estimation

- A general parametric method for density estimation method
- Estimate a set of model parameters that maximizes the likelihood.





Maximum-Likelihood Estimation

- Data: $D = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$
- Parameters: $\boldsymbol{\theta}$
- Model: $p(\boldsymbol{x}|\boldsymbol{\theta})$
- Likelihood of $\boldsymbol{\theta}$ with respective to a set of data samples

$$L(\boldsymbol{\theta}; D) = p(D|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\boldsymbol{x}_i|\boldsymbol{\theta})$$

- Maximum likelihood principle: find $\widehat{\boldsymbol{\theta}}$ that maximizes L
  - Agrees the most with the observation of current dataset

Log-likelihood function

- log-likelihood function

$$l(\boldsymbol{\theta}) \equiv \ln L(\boldsymbol{\theta}) = \ln p(D|\boldsymbol{\theta}) = \sum_i \ln p(x_i|\boldsymbol{\theta})$$

- Maximize likelihood function is equivalent to maximize log-likelihood function

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta})$$

$$\Rightarrow \nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = 0$$

$$\nabla_{\boldsymbol{\theta}} \equiv \begin{bmatrix} \frac{\partial}{\partial \theta_1} \\ \vdots \\ \frac{\partial}{\partial \theta_p} \end{bmatrix}$$

Example: Coin Flipping

- Estimate the probability π of getting a head upon flipping a coin
- Data:
  - Flip the coin "independently" 10 times ➔ sample $n = 10$ times
  - HHTHHHTTHH

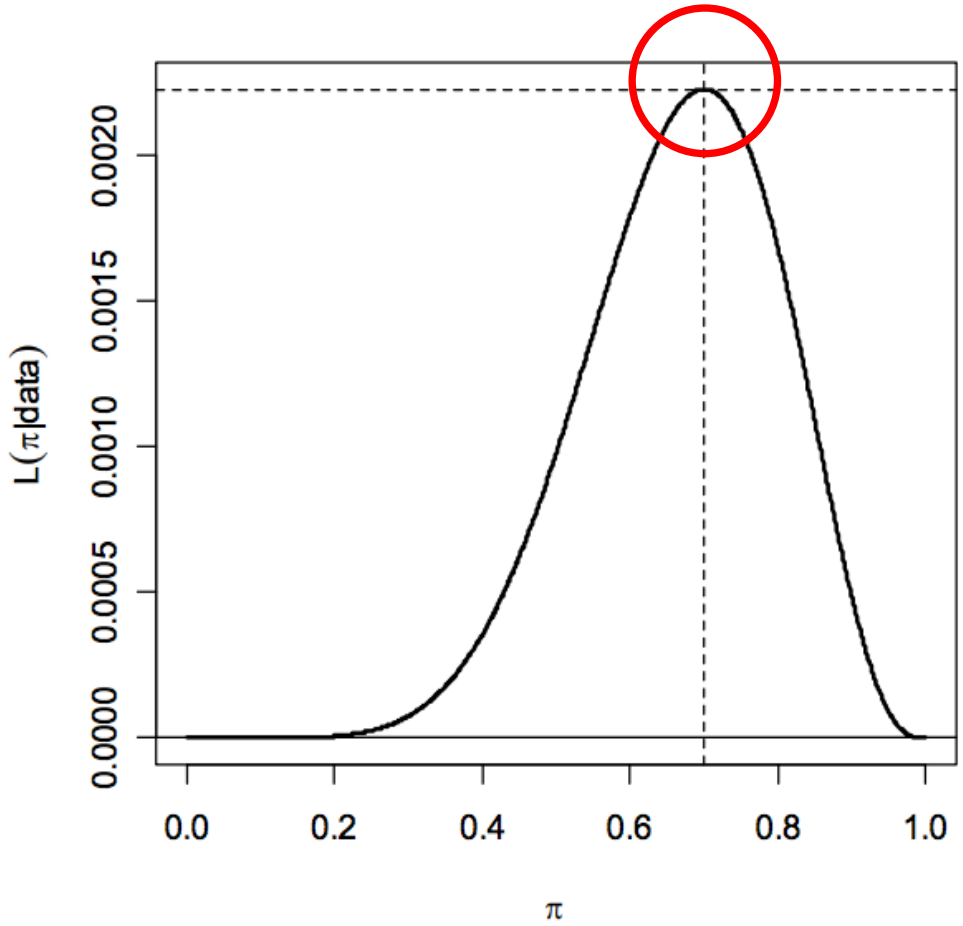
$$P(data \mid \theta) = P(\, HHTHHHTTHH \mid \pi\,)$$
$$= \pi\pi(1-\pi)\pi\pi\pi(1-\pi)(1-\pi)\pi\pi$$
$$= \pi^7\,(1-\pi)^3$$

- Likelihood function:

$$L(\pi \mid data) = L(\,\pi \mid HHTHHHTTHH\,)$$
$$= \pi^7\,(1-\pi)^3$$

Example: Coin Flipping (Cont'd)

π	$L(\pi\|\text{data}) = \pi^7(1-\pi)^3$
0.0	0.0
.1	.0000000729
.2	.00000655
.3	.0000750
.4	.000354
.5	.000977
.6	.00179
.7	.00222
.8	.00168
.9	.000478
1.0	0.0



Example: Coin Flipping (Cont'd)

- More generally, for n independent flips with x heads and $n - x$ tails

$$L(\pi \mid data) = \pi^x (1 - x)^{n-x}$$

- We want to maximize $L(\pi \mid data)$ or $L(\pi)$

$$\log L(\pi) = x \log \pi + (n - x) \log(1 - \pi)$$

- Differentiating $\log L(\pi)$ with repect to π

$$\frac{d \log L(\pi)}{d\pi} = \frac{x}{\pi} + (n - x) \frac{1}{1 - \pi} (-1) = \frac{x}{\pi} - \frac{n - x}{1 - \pi}$$

- Set the derivative to 0, we have

$$\hat{\pi} = \frac{x}{n}$$

The Gaussian Case: Unknown Mean

- Consider 1-d Gaussian Distribution

$$x_i \sim N(\mu, \sigma^2)$$

where σ^2 is known, i.e., $\theta = \mu$

$$p(x_i|\mu) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- The log-likelihood is then

$$l(\mu) = \sum_i \ln p(x_i|\mu) = \sum_i (-\frac{1}{2}\ln(2\pi\sigma^2) - \frac{(x_i-\mu)^2}{2\sigma^2})$$

- The MLE estimator for μ is then
  - $\nabla_\mu l(\mu) = 0 \Rightarrow \sum_i (x_i - \hat{\mu}) = 0 \Rightarrow \hat{\mu} = \frac{1}{n}\sum_i x_i$

The Gaussian Case: Unknown Mean & Variance

- **Consider 1-d Gaussian Distribution**

$$x_i \sim N(\mu, \sigma^2)$$

where both μ and σ^2 are unknown, i.e., $\boldsymbol{\theta} = (\mu, \sigma^2)$

$$p(x_i|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- **The log-likelihood is then**

$$l(\mu, \sigma^2) = \sum_i \ln p(x_i|\mu, \sigma^2) = \sum_i (-\frac{1}{2}\ln(2\pi\sigma^2) - \frac{(x_i - \mu)^2}{2\sigma^2})$$

- **The MLE estimators for μ and σ^2 are then**
  - $\frac{\partial l(\mu,\sigma^2)}{\partial \mu} = 0 \Rightarrow \sum_i (x_i - \hat{\mu})/\sigma^2 = 0 \Rightarrow \hat{\mu} = \frac{1}{n}\sum_i x_i$
  - $\frac{\partial l(\mu,\sigma^2)}{\partial \sigma^2} = 0 \Rightarrow \sum_i (-\frac{1}{2\widehat{\sigma^2}} + \frac{(x_i-\hat{\mu})^2}{2(\widehat{\sigma^2})\text{^}2}) = 0 \Rightarrow \widehat{\sigma^2} = \frac{1}{n}\sum_i (x_i - \hat{\mu})^2$

Note it is biased

References

- http://www.cs.princeton.edu/courses/archive/spr08/cos424/slides/clustering-1.pdf
- http://www.cs.princeton.edu/courses/archive/spr08/cos424/slides/clustering-2.pdf
- http://csc.csudh.edu/btang/seminar/slides/DBSCAN.pdf