LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Introduction to Machine Learning

Clustering

Department of Statistics – LMU Munich

Motivation

SUPERVISED VS. UNSUPERVISED LEARNING

Supervised Machine Learning:

- Supervised machine learning deals with **labeled** data, i.e., we have input data **x** and the outcome y of past events.
- Here, the aim is to learn relationships between **x** and y.

Unsupervised Machine Learning:

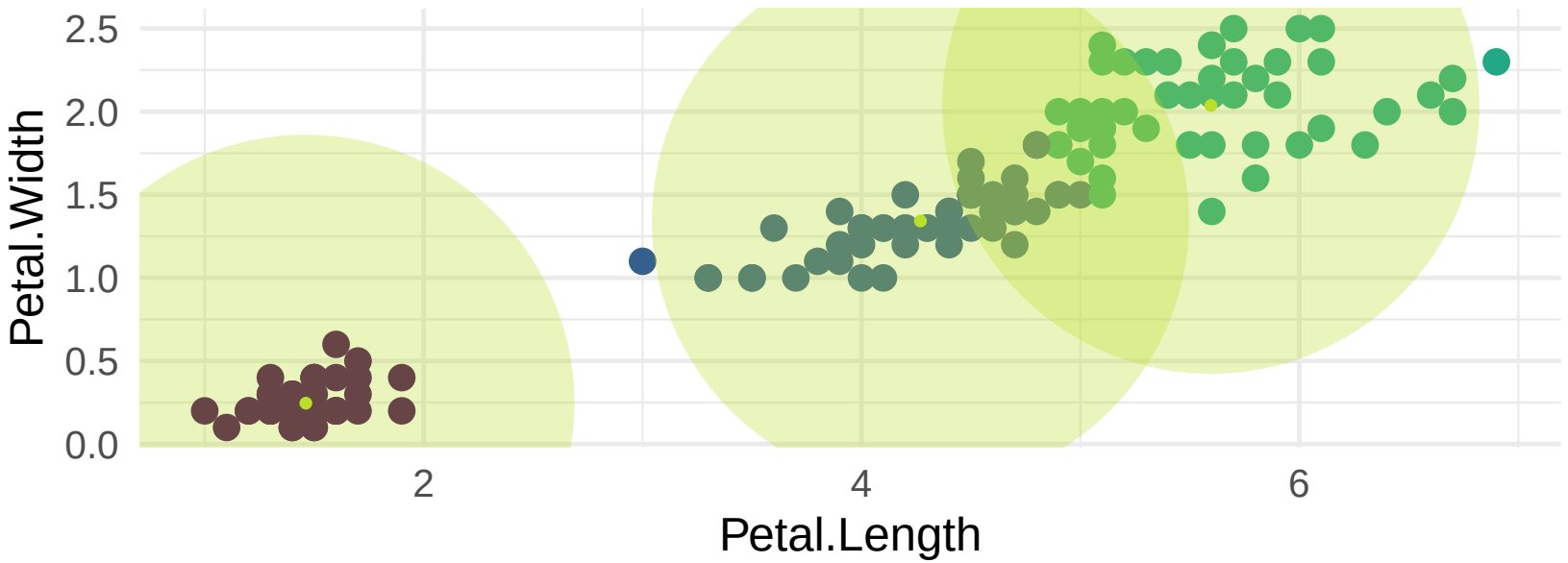
- Unsupervised machine learning deals with data that is **unlabeled**, i.e., there is no real output y.
- Here, the aim is to search for patterns within the inputs **x**.

MOTIVATION FOR CLUSTERING

Consider multivariate data with n observations (e.g. customers) and p features (e.g. characteristics of customers).

Task: divide data into groups (clusters), such that

- the observations in each cluster are as "similar" as possible (homogeneity within each cluster), and
- the clusters are as "far away" as possible from other clusters (heterogeneity between different clusters).



CLUSTERING VS. CLASSIFICATION

- In classification, the groups are known and we try to learn what differentiates these groups (i.e., learn a classification function) to properly classify future data.
- In clustering, we look at data, where groups are unknown and try to find similar groups.

Why do we need clustering?

- Discovery: looking for new insights in the data (e.g. finding groups of customers that buy a similar product).
- Derive a reduced representation of the full data set.

CLUSTERING: CUSTOMER SEGMENTATION

- In marketing, customer segmentation is an important task to understand customer needs and to meet with customer expectations.
- Customer data is partitioned in terms of similarities and the characteristics of each group are summarized.
- Marketing strategies are designed and prioritized according to the group size.

Hierarchical Clustering

HIERARCHICAL CLUSTERING

Hierarchical clustering is a recursive process that builds a hierarchy of clusters. We distinguish between:

1. Agglomerative (or bottom-up) clustering:
   - Start: Each observations is an **individual cluster**.
   - Repeat: Merge the two closest clusters.
   - Stop when there is only one cluster left.
2. Divisive (or top-down) clustering:
   - Start: All observations are within **one** cluster.
   - Repeat: Divide the cluster that results in two clusters with biggest distance.
   - Stop when each observation is an individual cluster.

HIERARCHICAL CLUSTERING

Let $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right\}$ be n observations of p features (dimensions), where $\mathbf{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_p^{(i)}\right)^T$. A data set $\mathcal{D}$ is a $(n \times p)$-matrix of the form:

	feature 1	...	...	feature p
$\mathbf{x}^{(1)}$	$x_1^{(1)}$	...	...	$x_p^{(1)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{x}^{(n)}$	$x_1^{(n)}$	...	...	$x_p^{(n)}$

HIERARCHICAL CLUSTERING

Hierarchichal clustering requires a definition for

- distances $d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$ between two observations $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$:
  - manhattan distance:

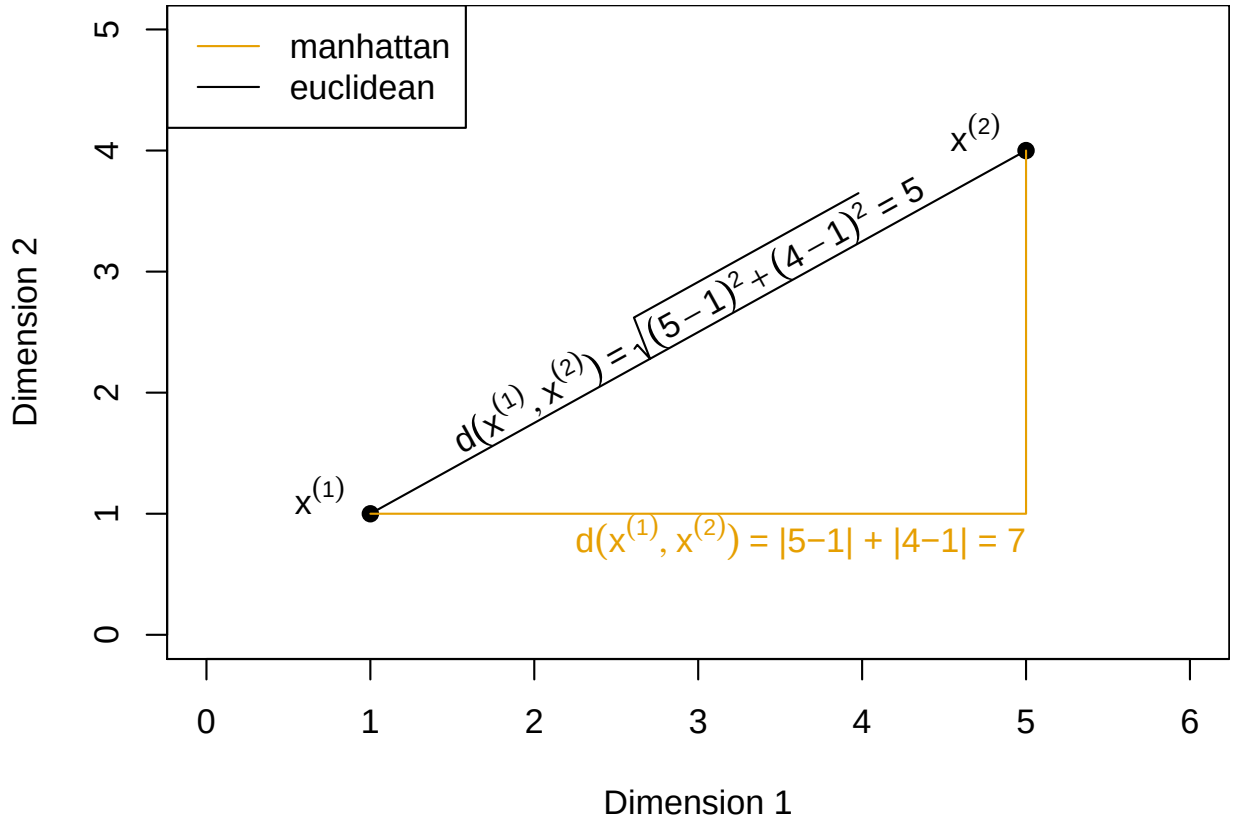
$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = ||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_1 = \sum_{k=1}^{p} \left| x_k^{(i)} - x_k^{(j)} \right|$$

  - euclidean distance:

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = ||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||_2 = \sqrt{\sum_{k=1}^{p} \left(x_k^{(i)} - x_k^{(j)} \right)^2}$$

- distances between two clusters (called linkage).

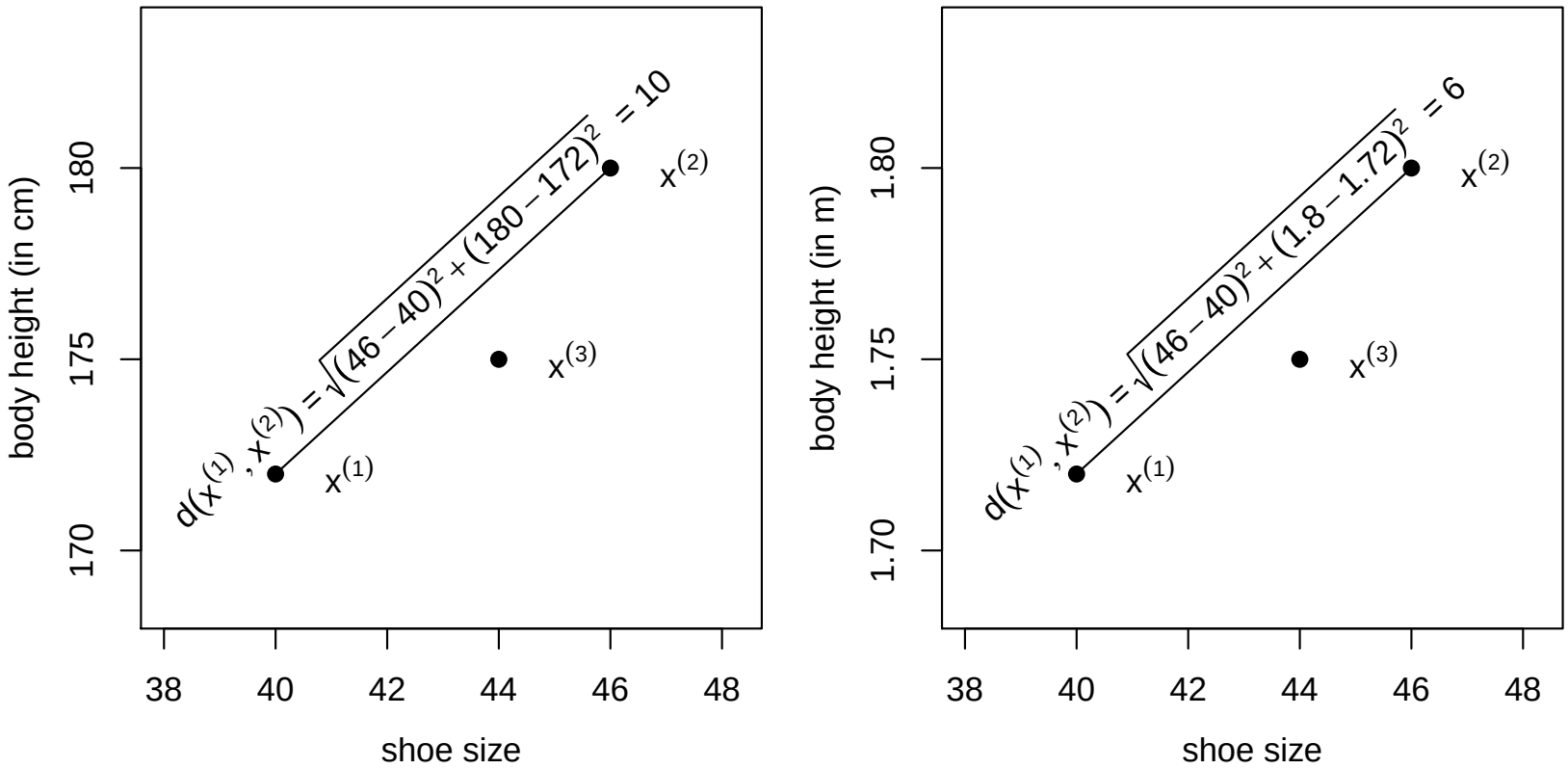
DISTANCES BETWEEN OBSERVATIONS



- manhattan: sum up the absolute distances in each dimension.
- euclidean: remember Pythagoras theorem from school?
- gower: can be used for mixed variables (categorical and numeric).

DISTANCES BETWEEN OBSERVATIONS

It is often a good idea to **normalize** the data before computing distances, especially when the scale of features is different, e.g.:



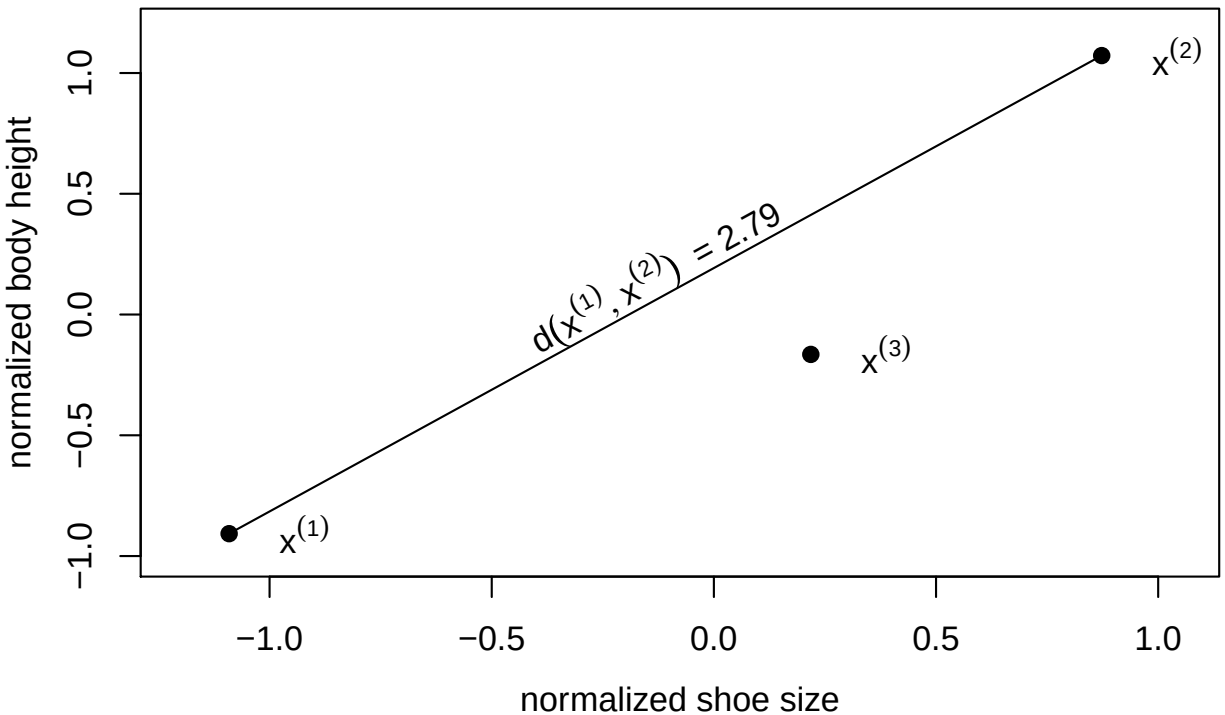
On the right plot, the distance is dominated by `shoe size`.

DISTANCES BETWEEN OBSERVATIONS

One possibility to normalize feature $\mathbf{x}_{\texttt{height}}$ is to compute

$$\tilde{\mathbf{x}}_{\texttt{height}} = \frac{\mathbf{x}_{\texttt{height}} - \text{mean}(\mathbf{x}_{\texttt{height}})}{\text{sd}(\mathbf{x}_{\texttt{height}})}.$$

Distances based on normalized data are better comparable and robust in terms of linear transformations (e.g. unit conversion).



DISTANCES BETWEEN CLUSTERS (LINKAGE)

- Assume that all observations $\left\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}\right\}$ belong to $k < n$ different clusters.
- For an arbitrary cluster j we define its **index space** by

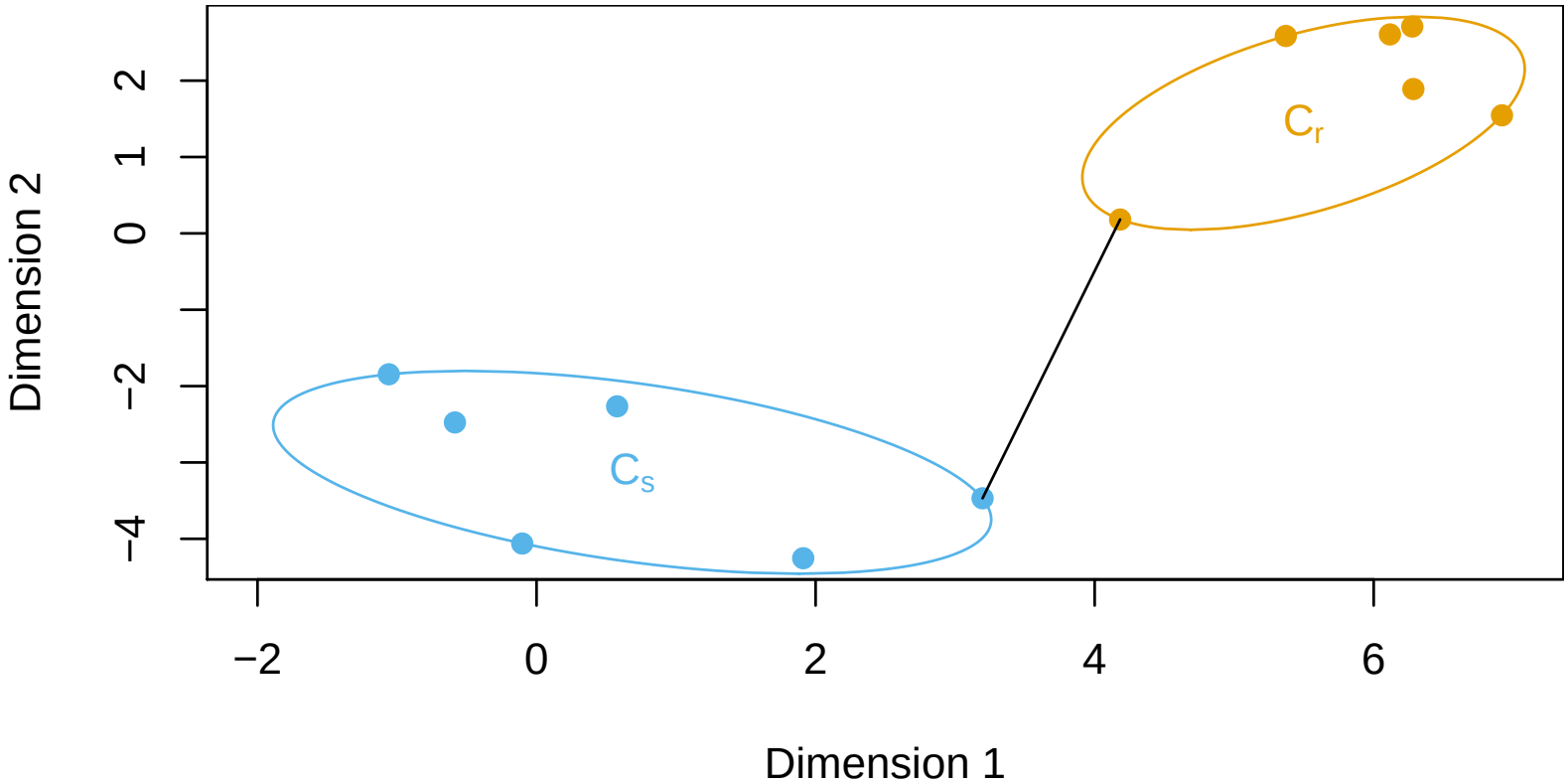
$$C_j := \{i \subset \{1, \ldots, k\} \mid \mathbf{x}^{(i)} \text{ belongs to cluster } j\}$$

  Let $n_j = |C_j|$ be the size of cluster j.
- The linkage of two clusters C_r and C_s is a "score" describing their distance.

The most popular and simplest linkages are

- Single Linkage
- Complete Linkage
- Average Linkage
- Centroid Linkage

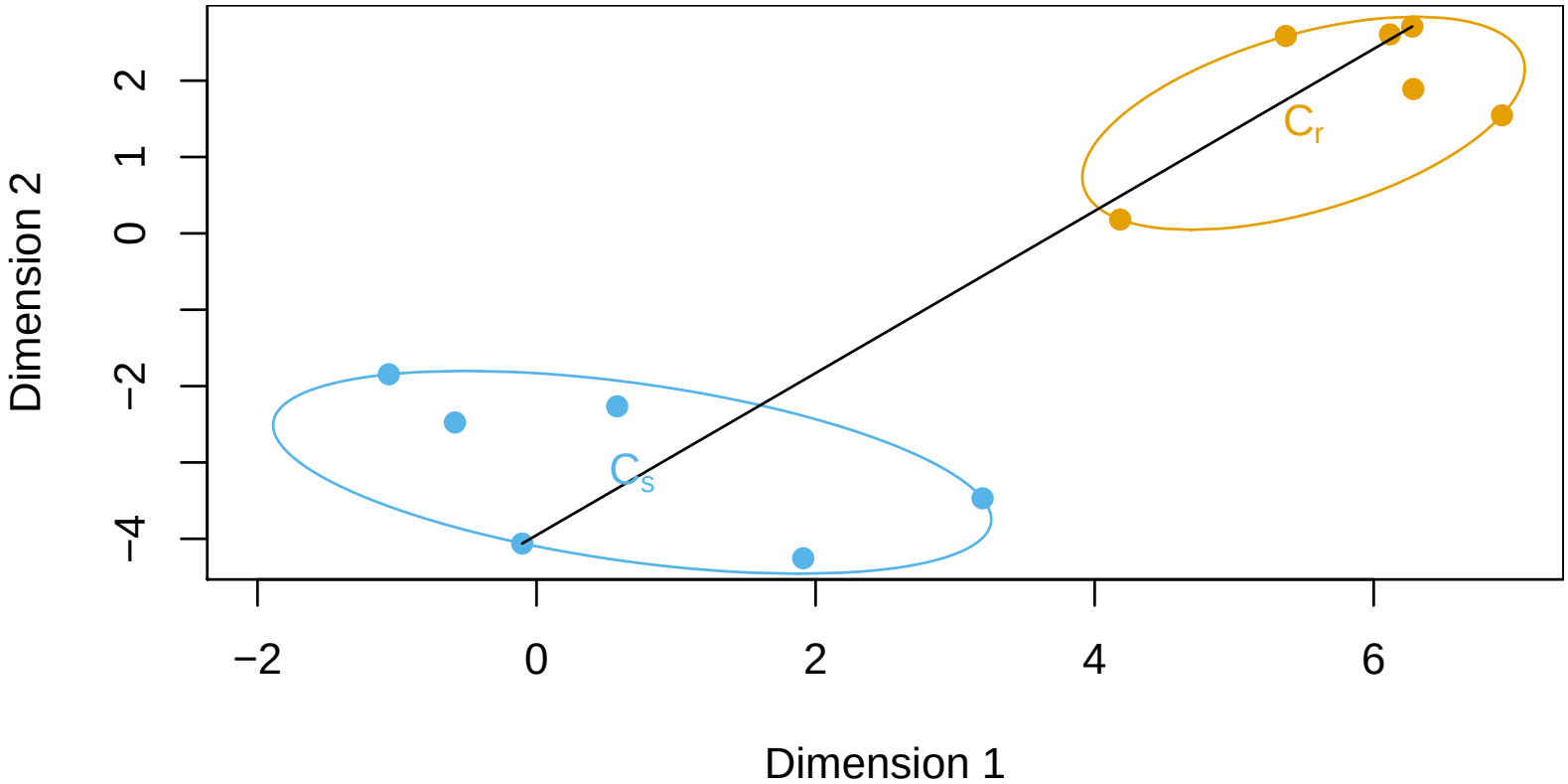
SINGLE LINKAGE



Single linkage defines the distance of the **closest point pairs** from different clusters as the distance between two clusters:

$$d_{\text{single}}(C_r, C_s) = \min_{i \in C_r, j \in C_s} d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

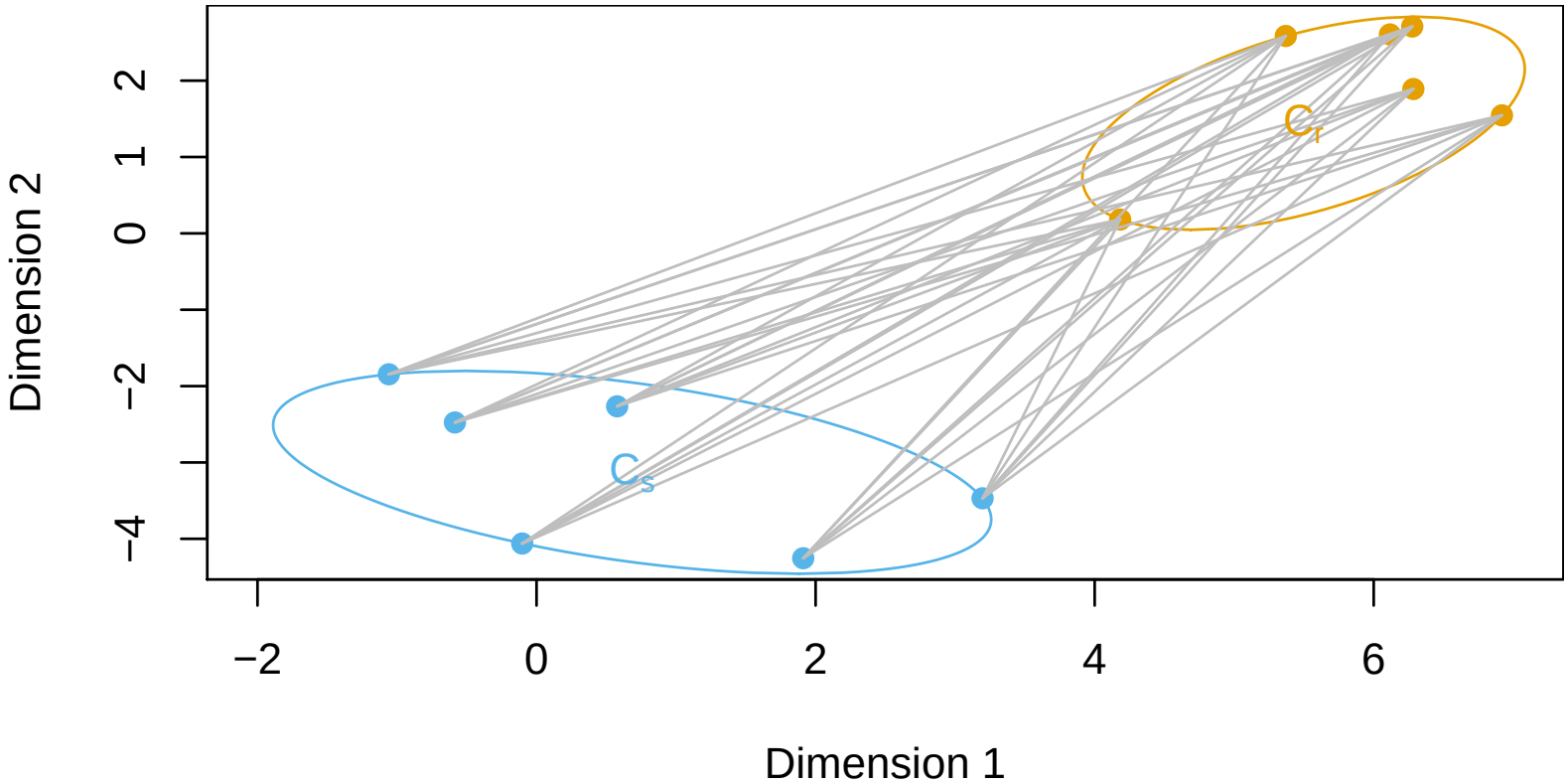
COMPLETE LINKAGE



Complete linkage defines the distance of the **furthest point pairs** of different clusters as the distance between two clusters:

$$d_{\text{complete}}(C_r, C_s) = \max_{i \in C_r, j \in C_s} d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

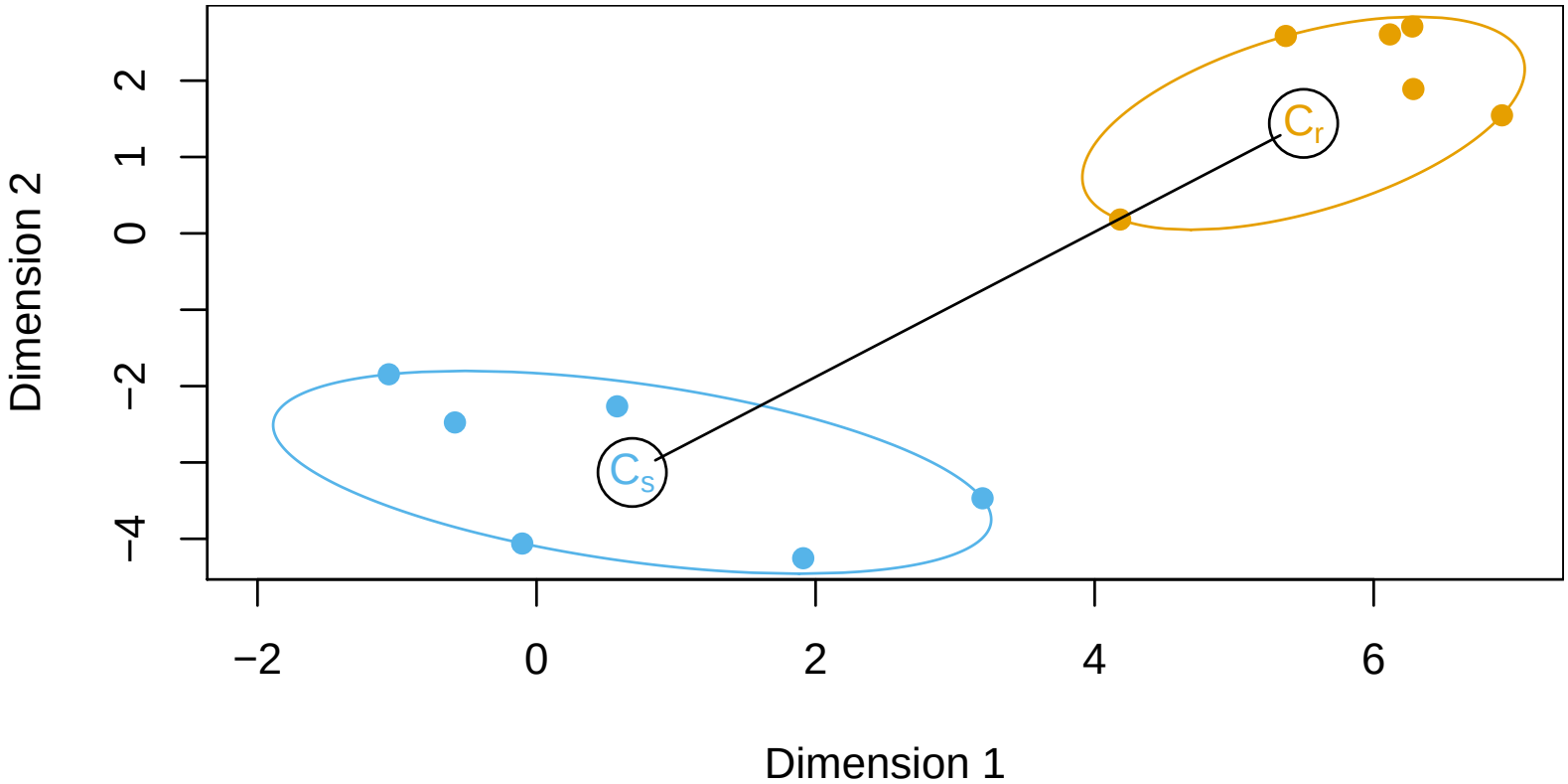
AVERAGE LINKAGE



In average linkage, the distance between two clusters is defined as the average distance across **all** pairs of two different clusters:

$$d_{\text{average}}(C_r, C_s) = \frac{1}{n_r n_s} \sum_{i \in C_r} \sum_{j \in C_s} d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

CENTROID LINKAGE



Centroid linkage defines the distance between two clusters as the distance between the two cluster centroids. The centroid of a cluster C_s with n_s points is the mean value of each dimension:

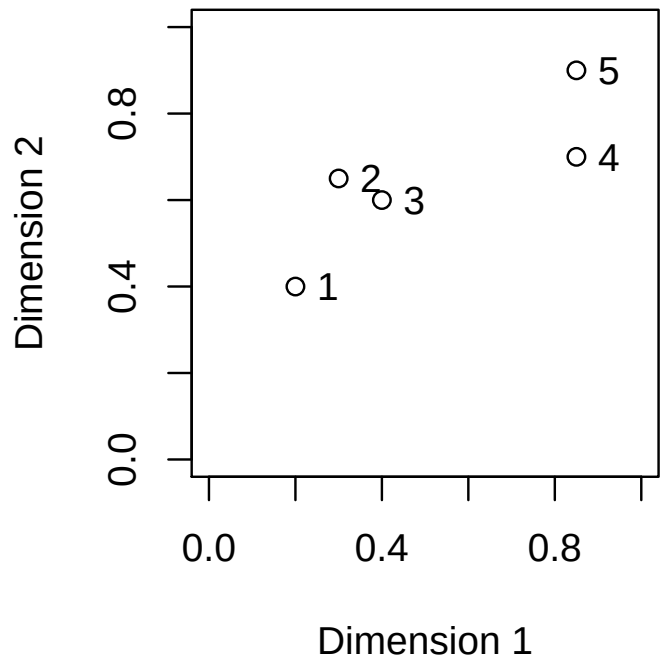
$$\bar{\mathbf{x}}_s = \frac{1}{n_s} \sum_{i \in C_s} \mathbf{x}^{(i)}$$

EXAMPLE: HIERARCHICAL CLUSTERING

Agglomerative hierarchical clustering starts with all points forming their own cluster and iteratively merges them until all points form a single cluster containing all points.

Example:

Step 1: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$

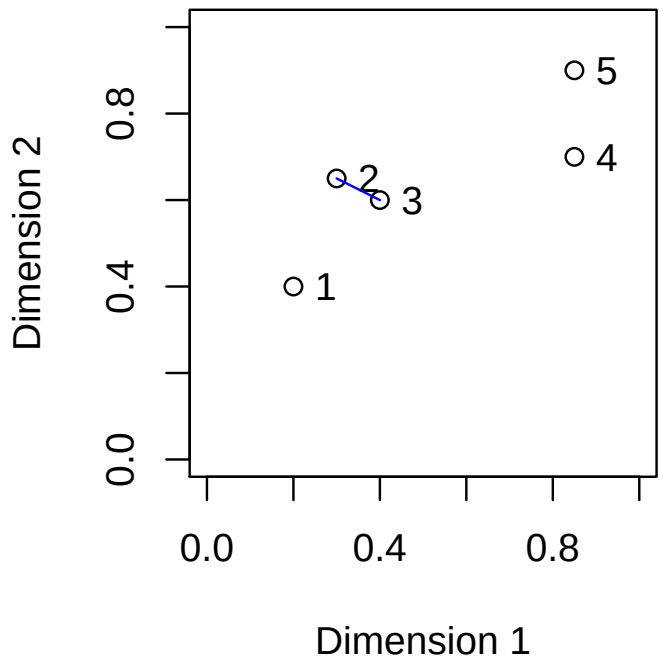


EXAMPLE: HIERARCHICAL CLUSTERING

Agglomerative hierarchical clustering starts with all points forming their own cluster and iteratively merges them until all points form a single cluster containing all points.

Example:

Step 1: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$
Step 2: $\{1\}, \{2, 3\}, \{4\}, \{5\}$

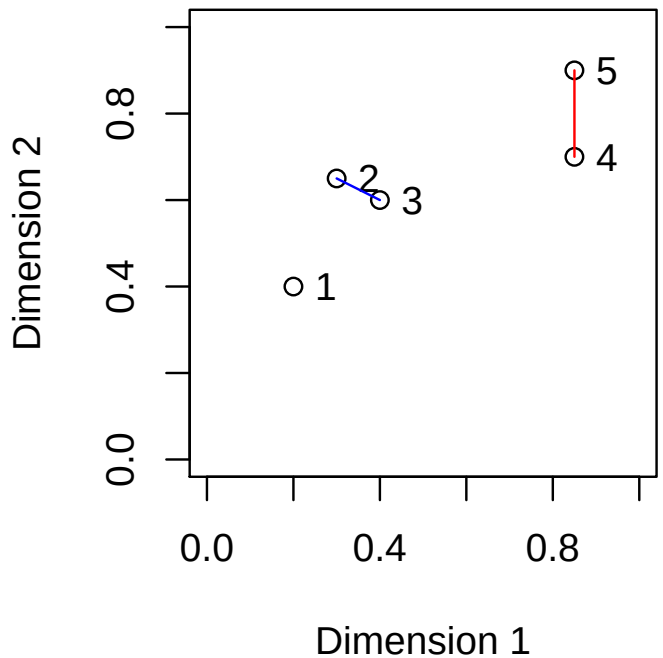


EXAMPLE: HIERARCHICAL CLUSTERING

Agglomerative hierarchical clustering starts with all points forming their own cluster and iteratively merges them until all points form a single cluster containing all points.

Example:

Step 1: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$
Step 2: $\{1\}, \{2, 3\}, \{4\}, \{5\}$
Step 3: $\{1\}, \{2, 3\}, \{4, 5\}$

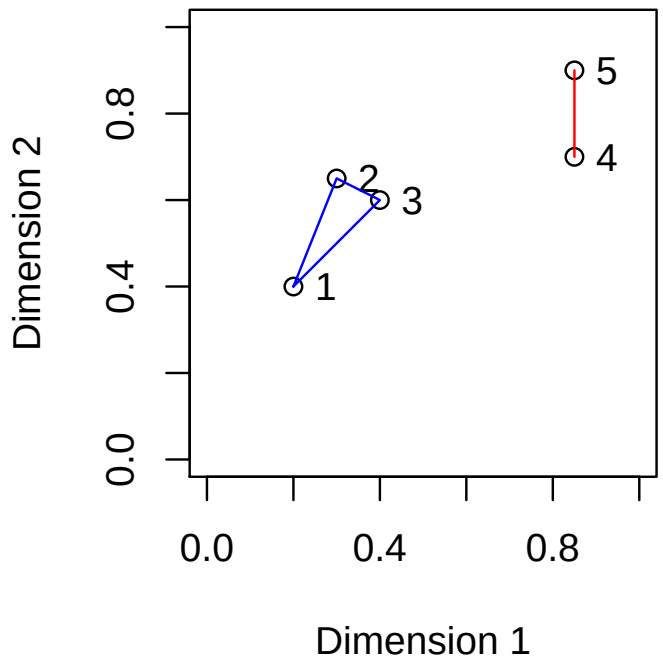


EXAMPLE: HIERARCHICAL CLUSTERING

Agglomerative hierarchical clustering starts with all points forming their own cluster and iteratively merges them until all points form a single cluster containing all points.

Example:

Step 1: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$
Step 2: $\{1\}, \{2, 3\}, \{4\}, \{5\}$
Step 3: $\{1\}, \{2, 3\}, \{4, 5\}$
Step 4: $\{1, 2, 3\}, \{4, 5\}$

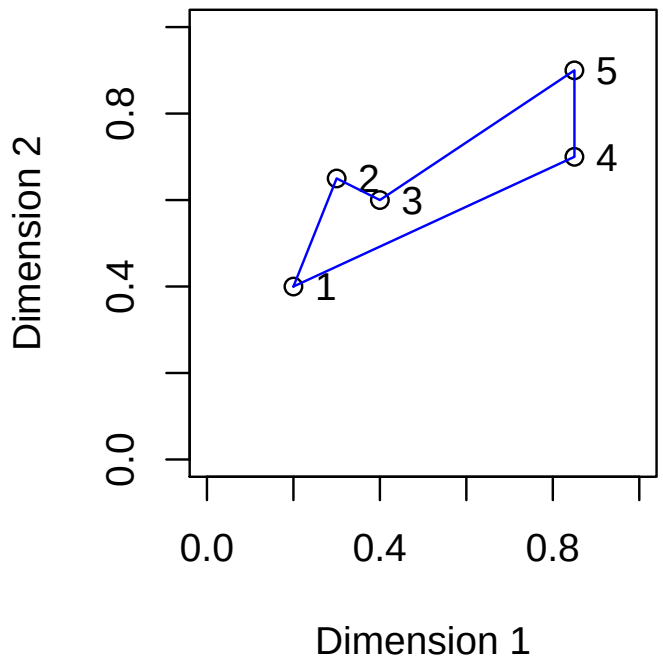


EXAMPLE: HIERARCHICAL CLUSTERING

Agglomerative hierarchical clustering starts with all points forming their own cluster and iteratively merges them until all points form a single cluster containing all points.

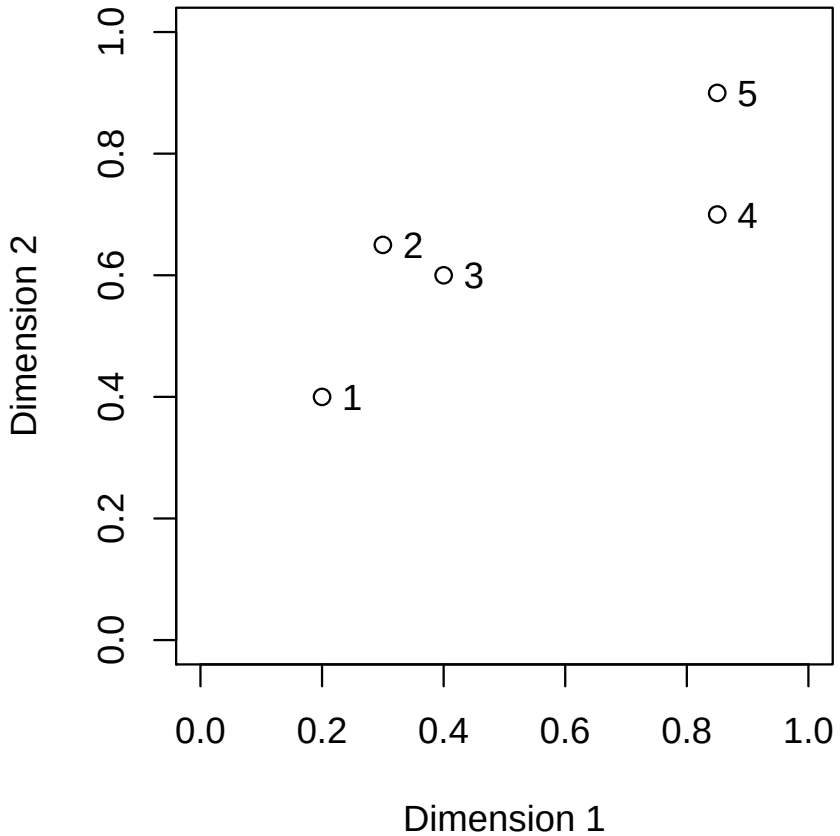
Example:

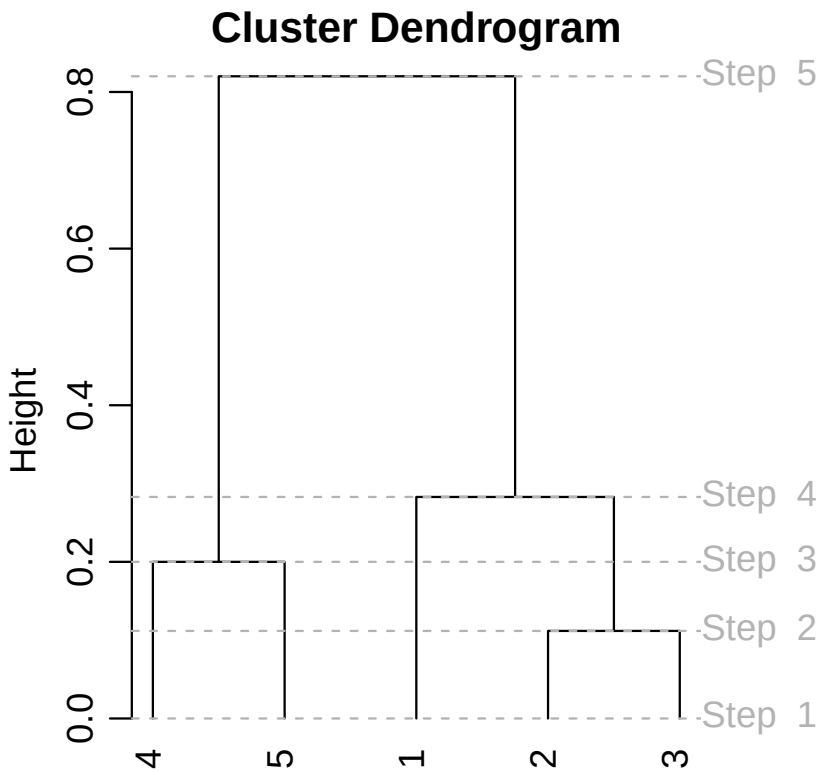
Step 1: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$
Step 2: $\{1\}, \{2, 3\}, \{4\}, \{5\}$
Step 3: $\{1\}, \{2, 3\}, \{4, 5\}$
Step 4: $\{1, 2, 3\}, \{4, 5\}$
Step 5: $\{1, 2, 3, 4, 5\}$



DENDROGRAM

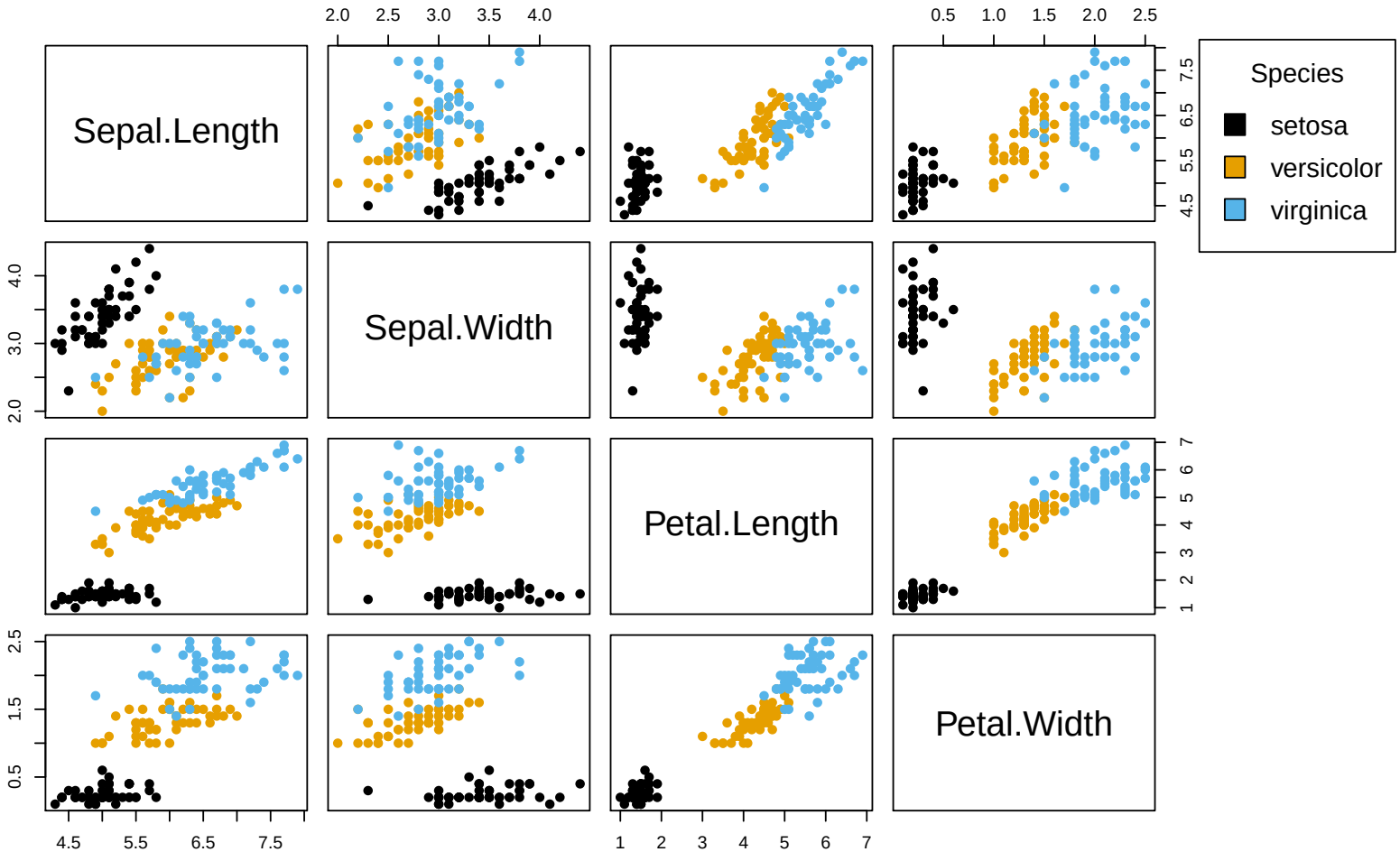
A dendrogram is a tree showing which clusters / observations are merged after each step. The “height” is proportional to the distance between the two merged clusters:





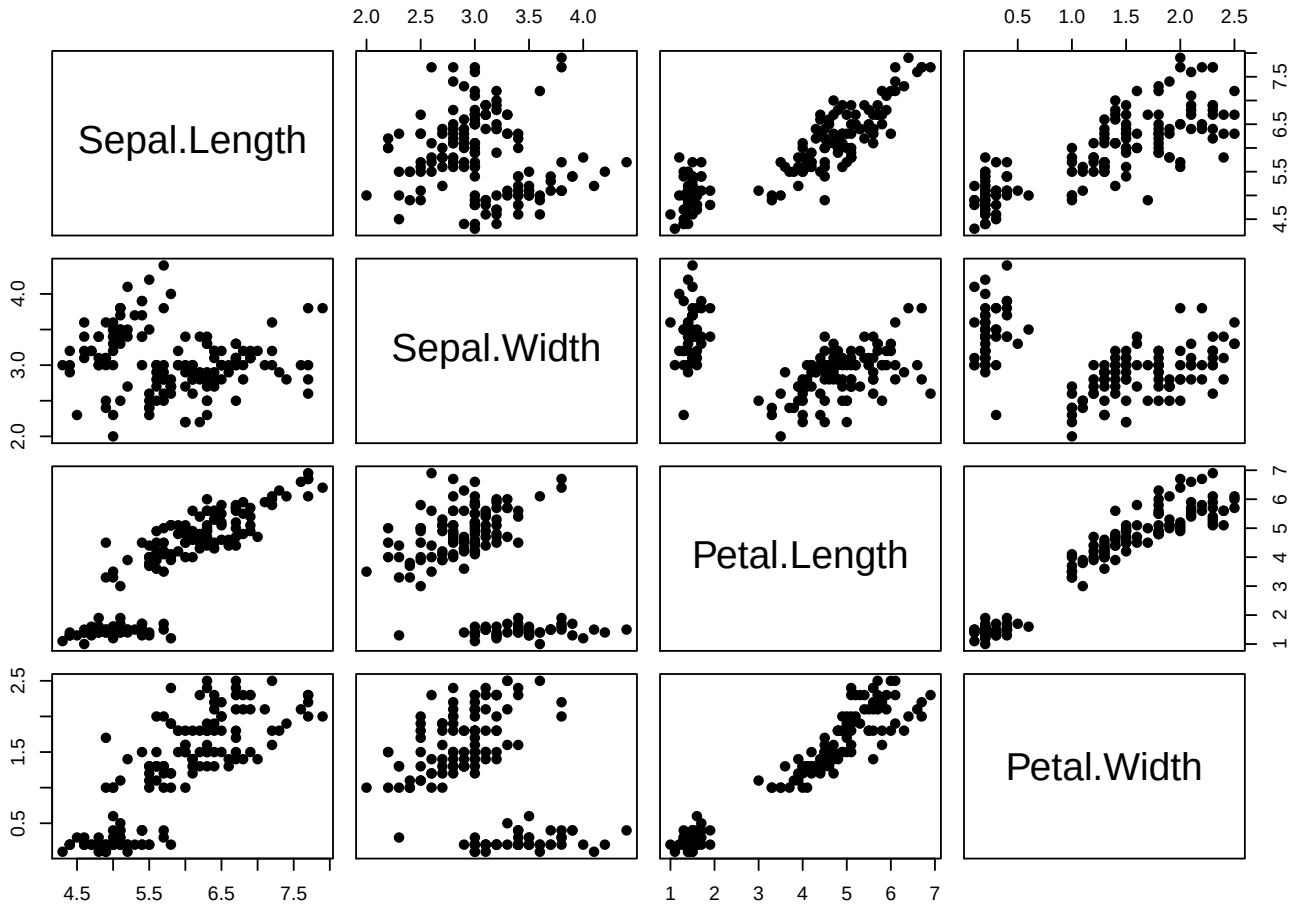
EXAMPLE: IRIS DATA

The data contains 150 leaf measurements for 3 flower species:

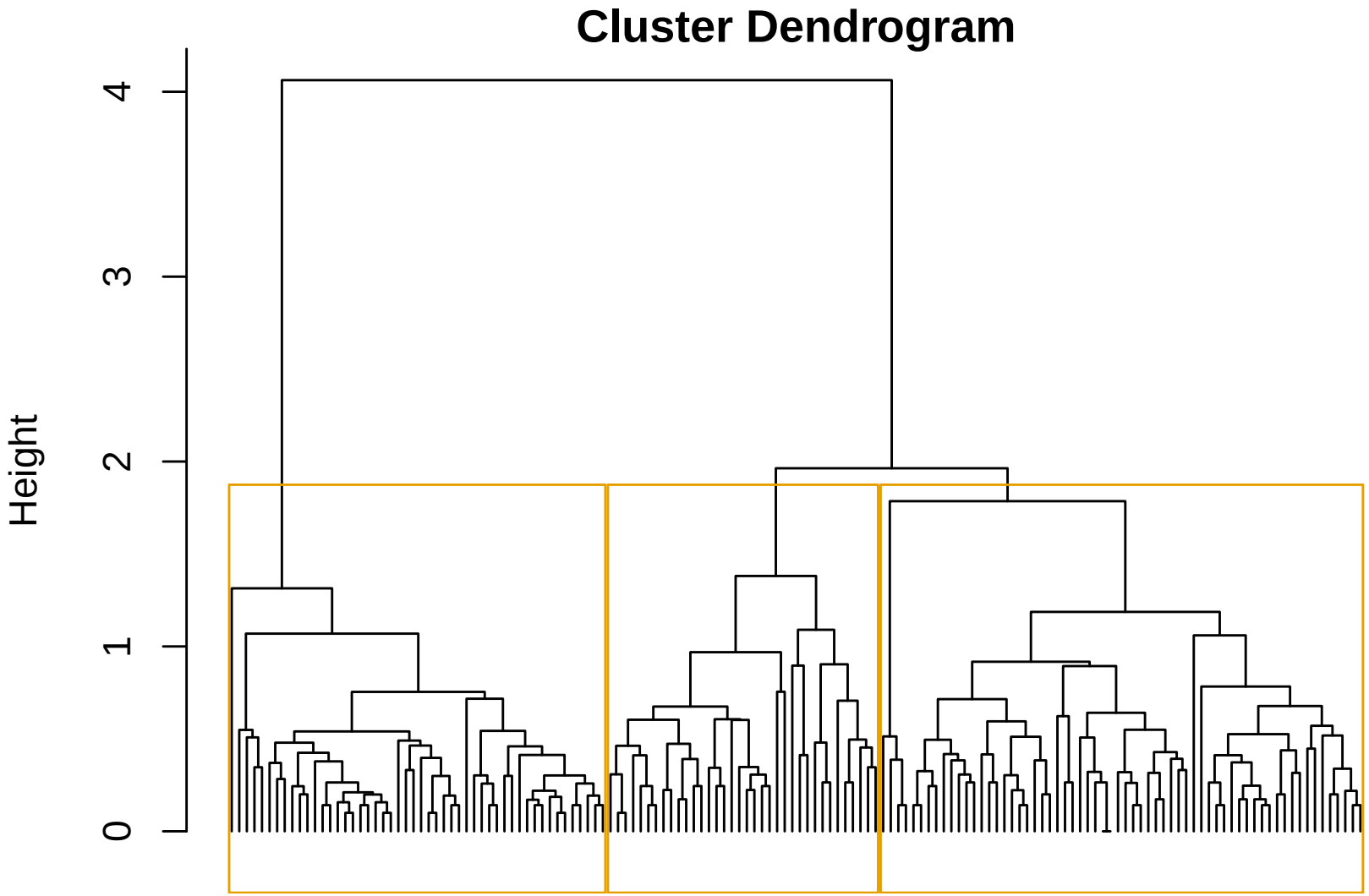


R EXAMPLE WITH IRIS DATA

We now “forget” the real groups specified by the `Species` variable and try to find clusters based on the leaf measurements.

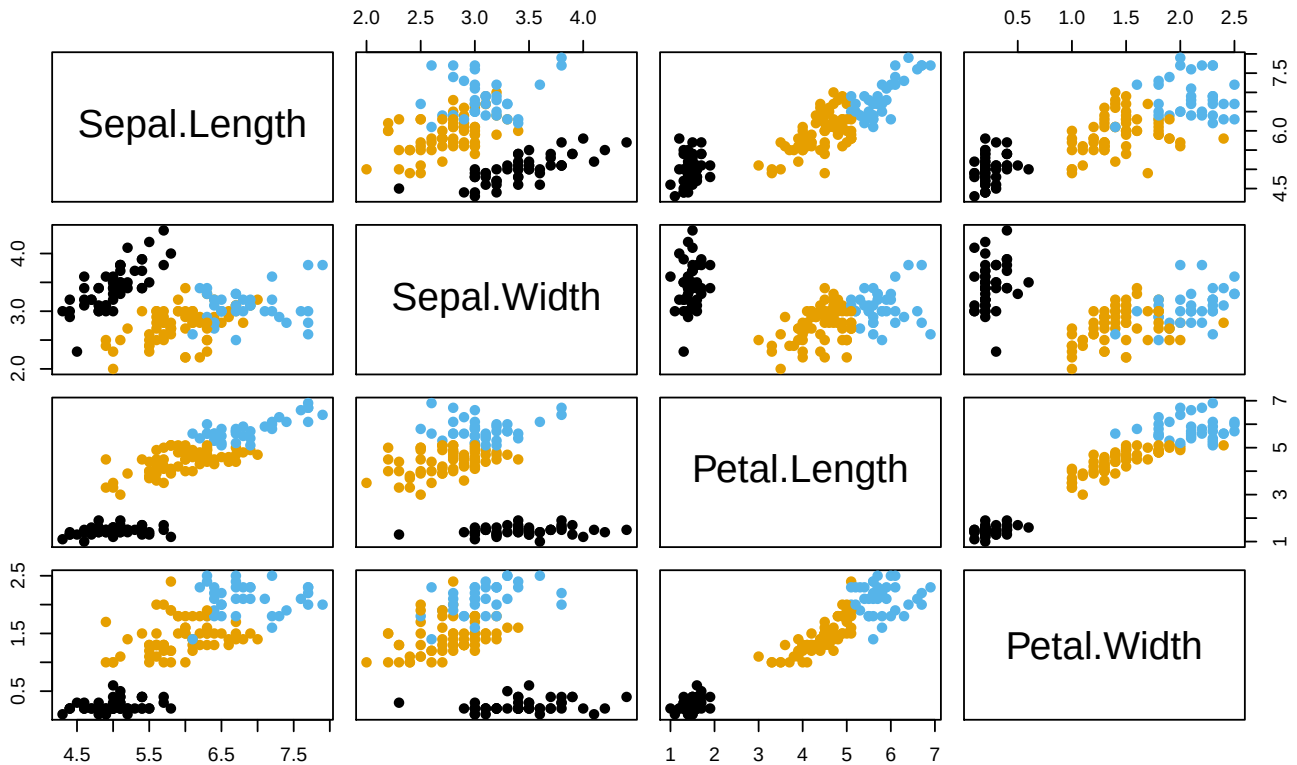


EXAMPLE: IRIS DATA



EXAMPLE: IRIS DATA

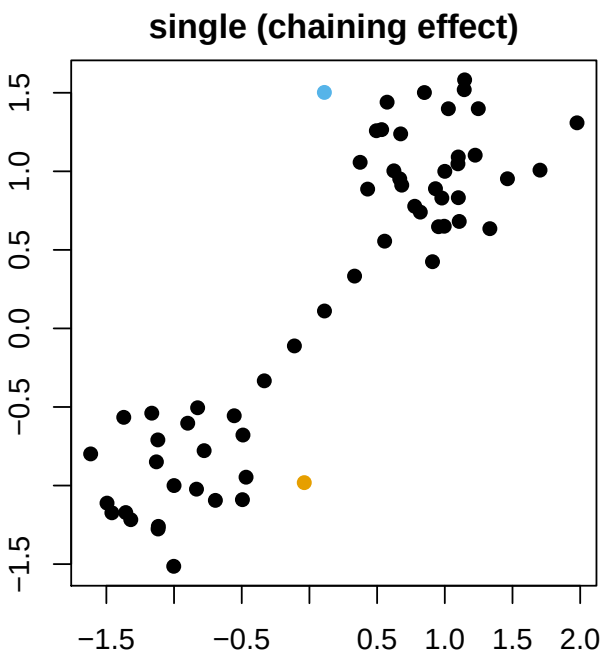
We can extract the clustering assignments by cutting the dendrogram, e.g. using $k = 3$ clusters:

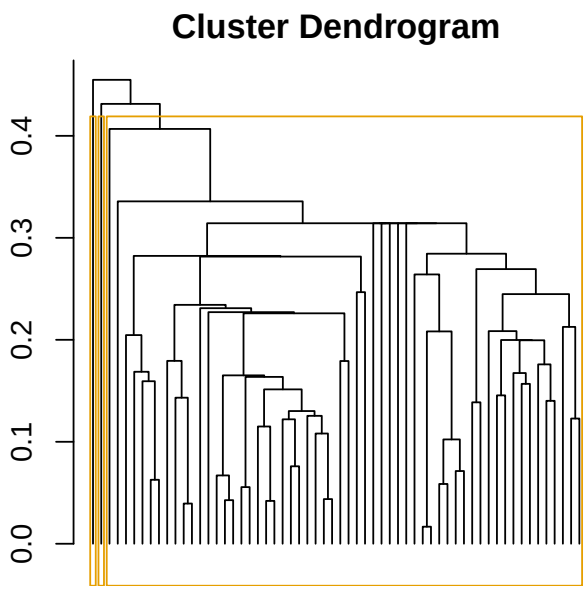


PROPERTIES: SINGLE LINKAGE

Single linkage introduces the **chaining problem**:

- Only one pair of points needs to be close to merge clusters.
- A chain of points can expand a cluster over long distances.
- Points within a cluster can be too widely spread and not dense enough.

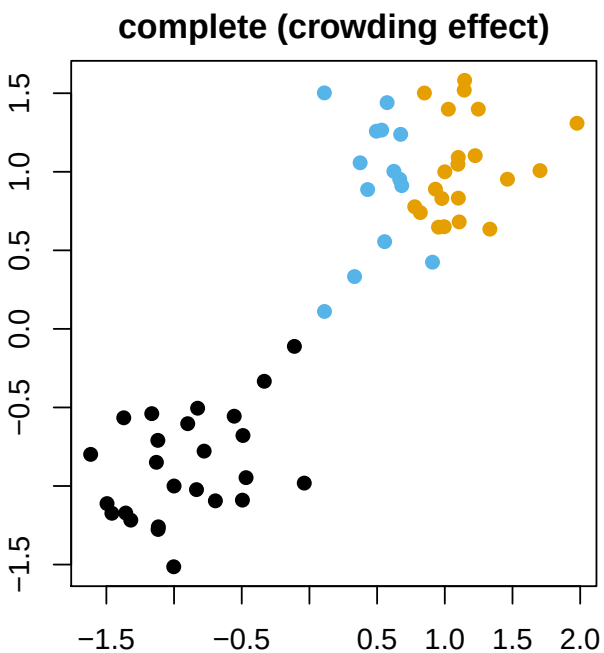


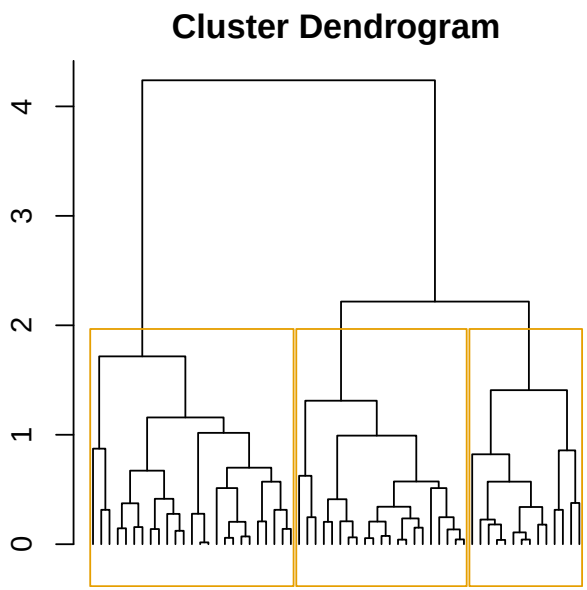


PROPERTIES: COMPLETE LINKAGE

Complete linkage avoids chaining, but suffers from **crowding**:

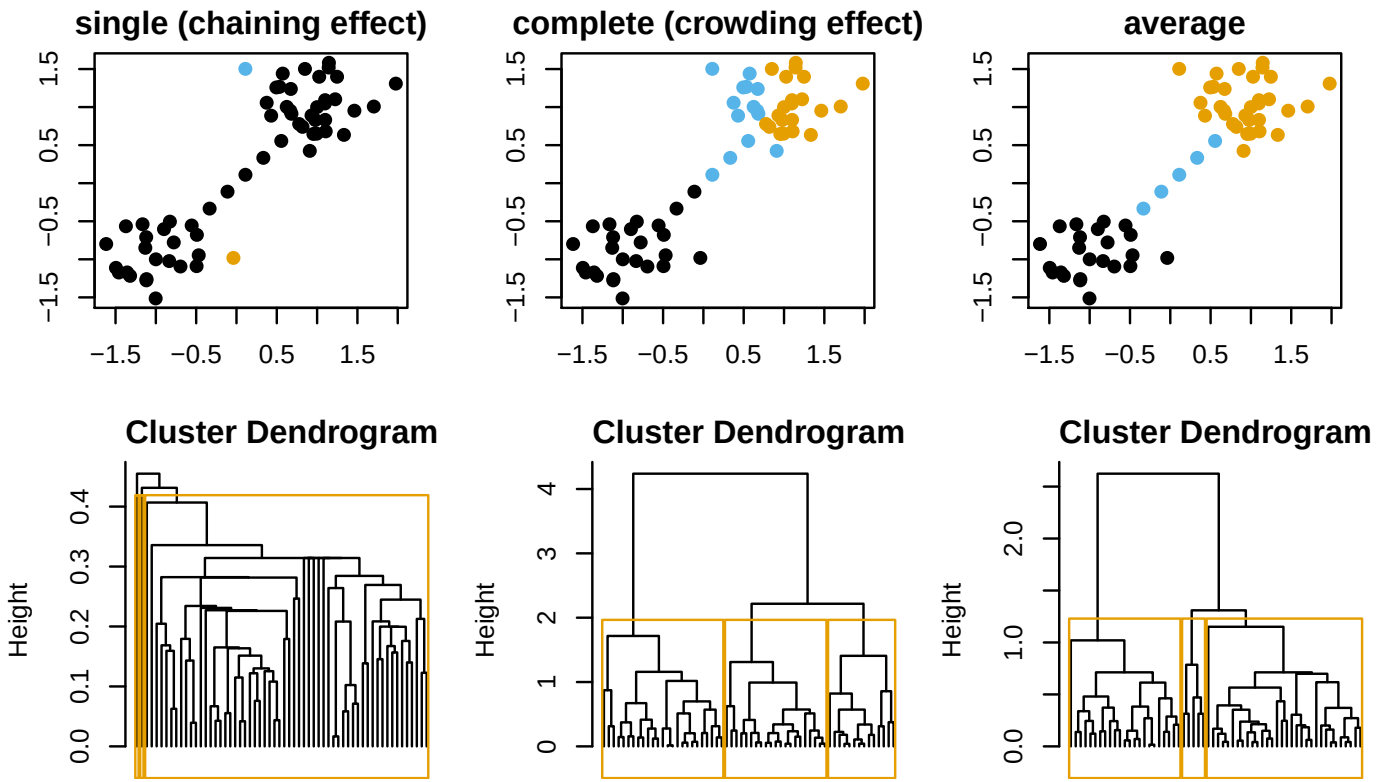
- Merging is based on the furthest distance of point pairs from different clusters.
- Points of two different clusters can thus be closer than points within a cluster.
- Clusters are dense, but too close to each other.





PROPERTIES: AVERAGE LINKAGE

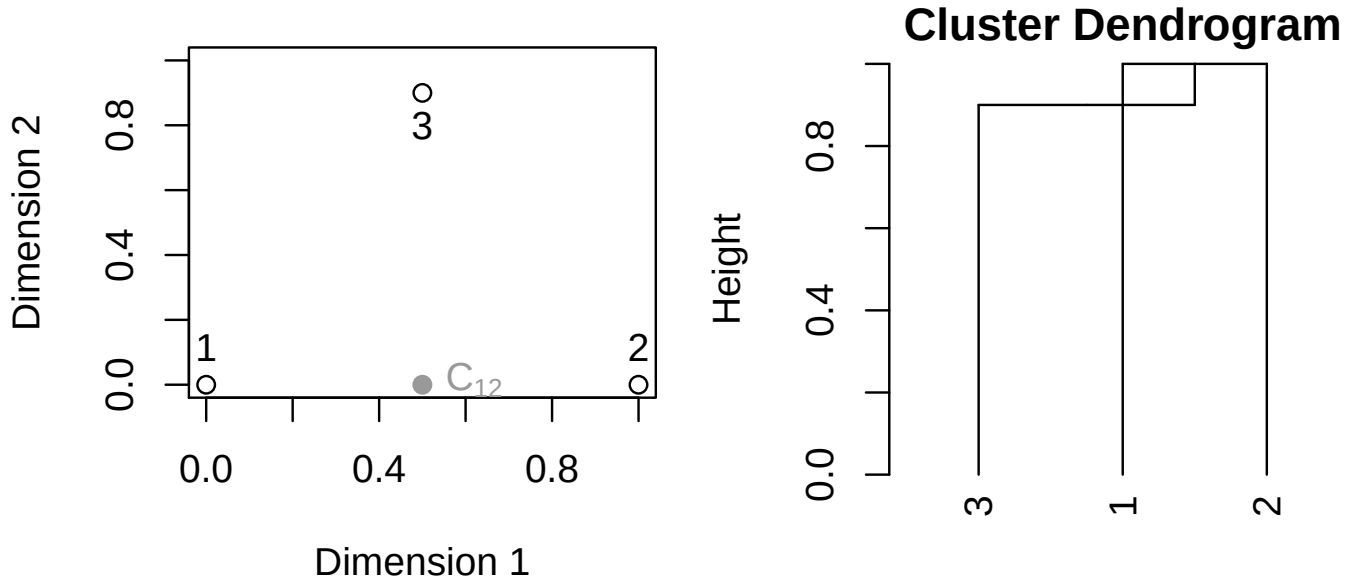
- Average linkage is based on the average distance between clusters and tries to avoid crowding and chaining.
- Produces clusters that are quite dense and rather far apart.



PROPERTIES: CENTROID LINKAGE

- Centroid linkage defines the distance based on **artificial data points** (the cluster centers), which produces dendrograms **with inversions**, i.e., the distance between the clusters to be merged can be smaller in the next step.
- In single, complete and average linkage, the distance between the clusters to be merged increases in each step. $\Rightarrow$ always produces dendrograms **without inversions**.

PROPERTIES: CENTROID LINKAGE



SUMMARY

- **Hierarchical agglomerative clustering methods** iteratively merge observations/clusters until all observations are in one single cluster.
- Results in a hierarchy of clustering assignments which can be visualized in a **dendrogram**. Each node of the dendrogram represents a cluster and its "height" is proportional to the distance of its child nodes.
- The most common linkage functions are **single, complete, average** and **centroid** linkage. There is no perfect linkage and each linkage has its own advantages and disadvantages.

Partitioning Clustering Methods

OPTIMAL PARTITIONING CLUSTERING

Hierarchical clustering:
Stepwise merging (agglomerative methods) or dividing (divisive methods) of clusters based on distances and linkages. The number of clusters are selected by splitting the dendrogram at a specific threshold for the "height" after visual inspection.

Partitioning clustering:
Partitions the n observations into a predefined number of k clusters by optimizing a numerical criterion. The most common partitioning methods are:

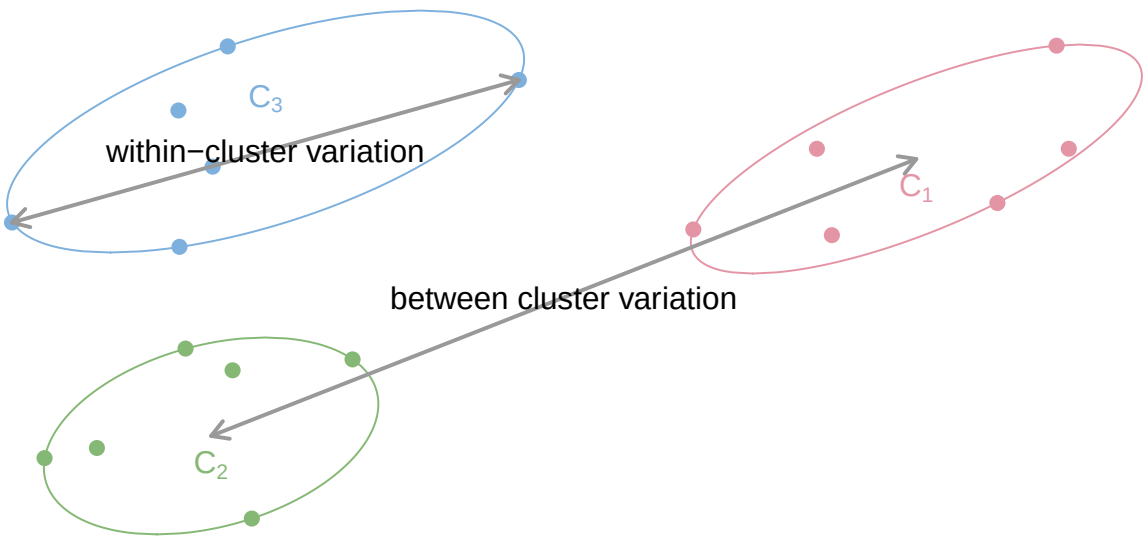
- k-means
- k-medians
- k-medoids (Partitioning Around Medoids (PAM))

K-MEANS

k-means partitions the n observations into k predefined clusters $C_1, C_2, \ldots, C_k$ by minimizing the **compactness**, i.e. the **within-cluster variation** of all clusters using

$$\sum_{j=1}^{k} \sum_{i \in C_j} \|\mathbf{x}^{(i)} - \bar{\mathbf{x}}_j\|_2^2 \to \min,$$

where $\bar{\mathbf{x}}_j = \frac{1}{n_j} \sum_{i \in C_j} \mathbf{x}^{(i)}$ is the centroid of cluster j and n_j is the number of observations in cluster j.



K-MEANS

Idea: Consider every possible partition of n observations into k clusters and select the one with the lowest **within-cluster variation**.

Problem: Requires trying all possible assignments of n observations into k clusters, which in practice is nearly impossible (Hothorn et al., 2009, p. 322):

n	k	Number of possible partitions
15	3	2.375.101
20	4	45.232.115.901
100	5	10^{68}

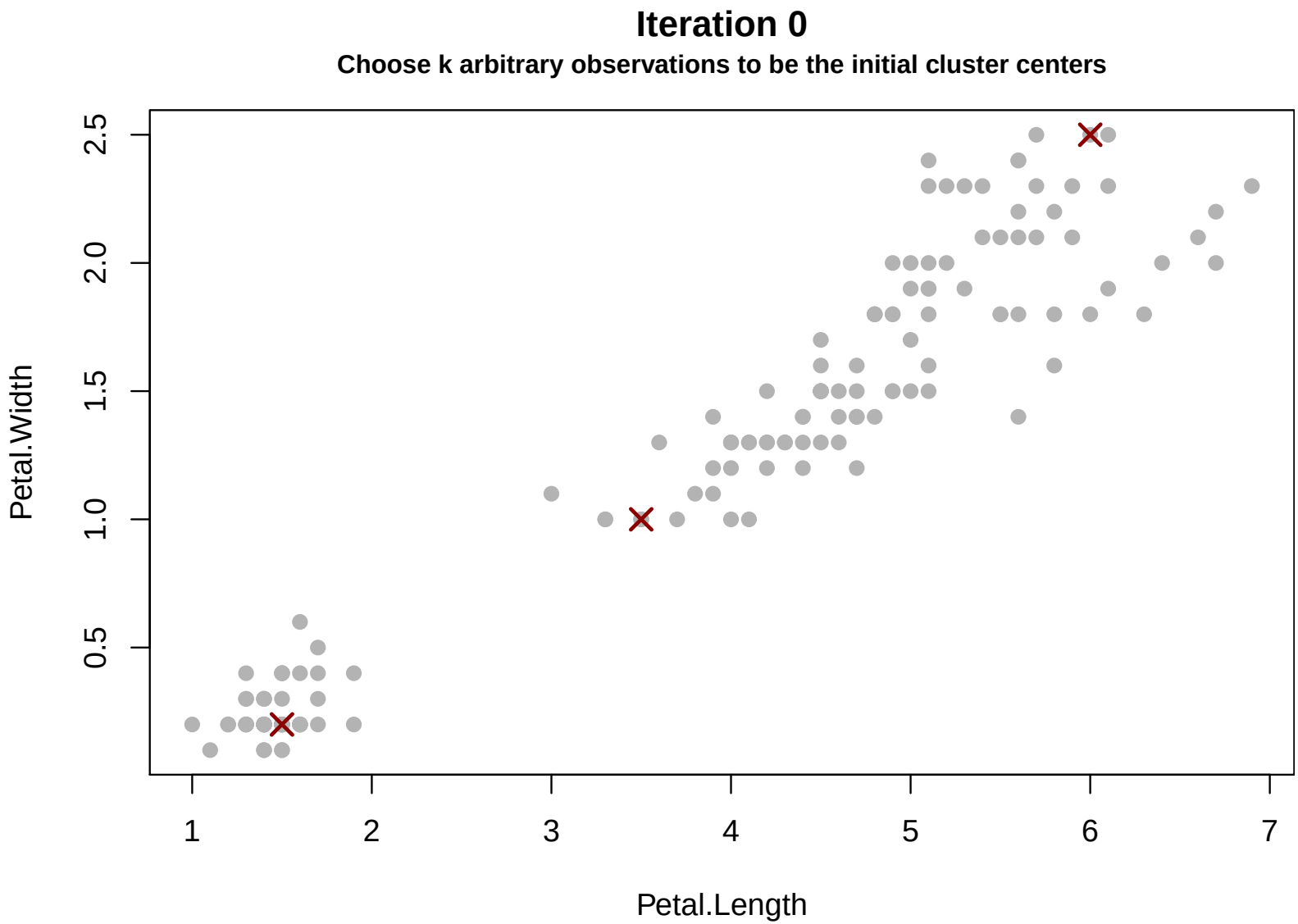
Hothorn, T., Everitt, B. S. (2009). A handbook of statistical analyses using R. Chapman and Hall/CRC.

K-MEANS

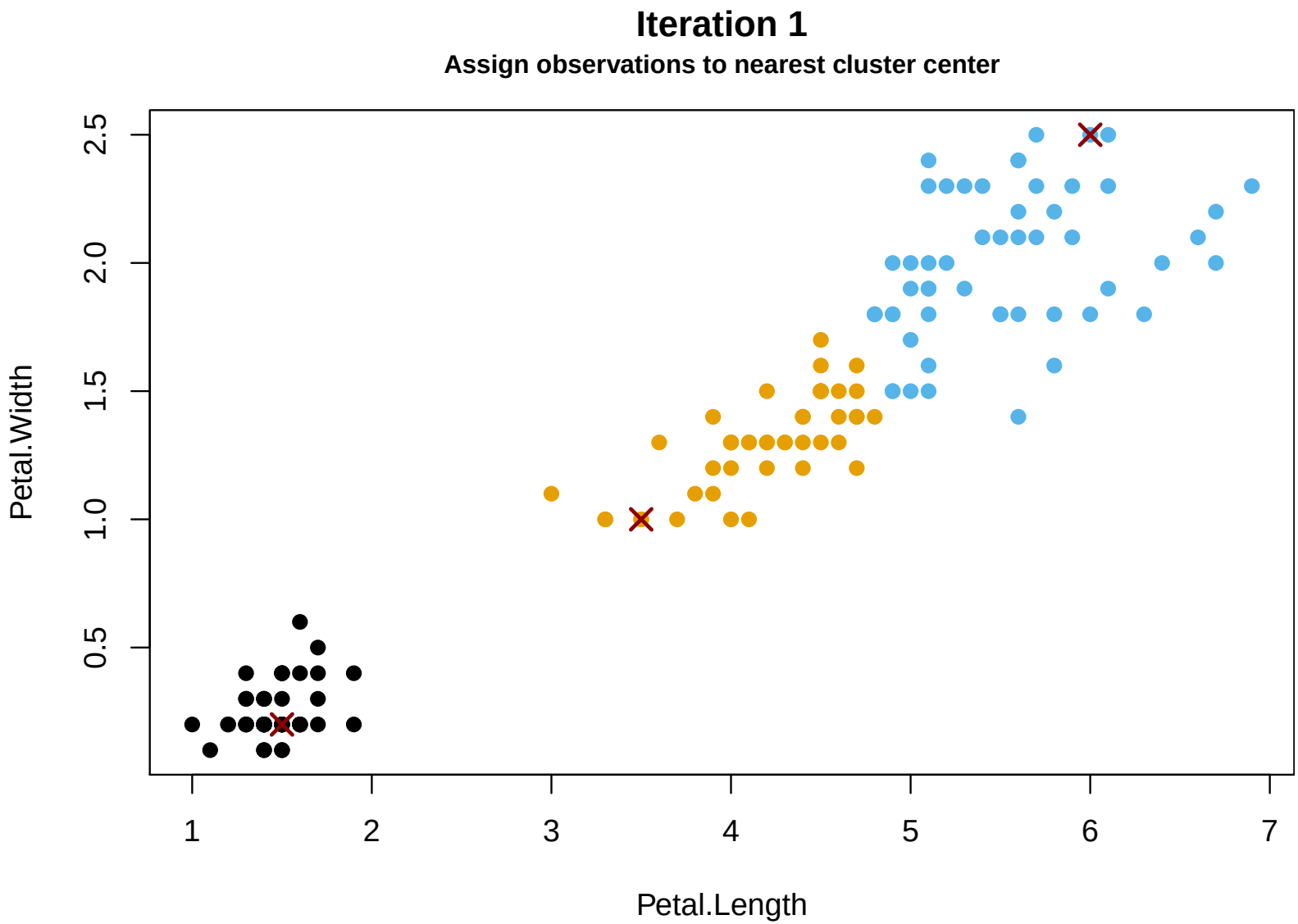
Use an approximation:

1. **Initialization:** Choose k arbitrary observations to be the initial cluster centers.
2. **Assignment:** Assign every observation to the cluster with the closest center.
3. **Update:** Compute the new center of each cluster as the mean of its members.
4. Repeat (2) and (3) until the centers do not move.

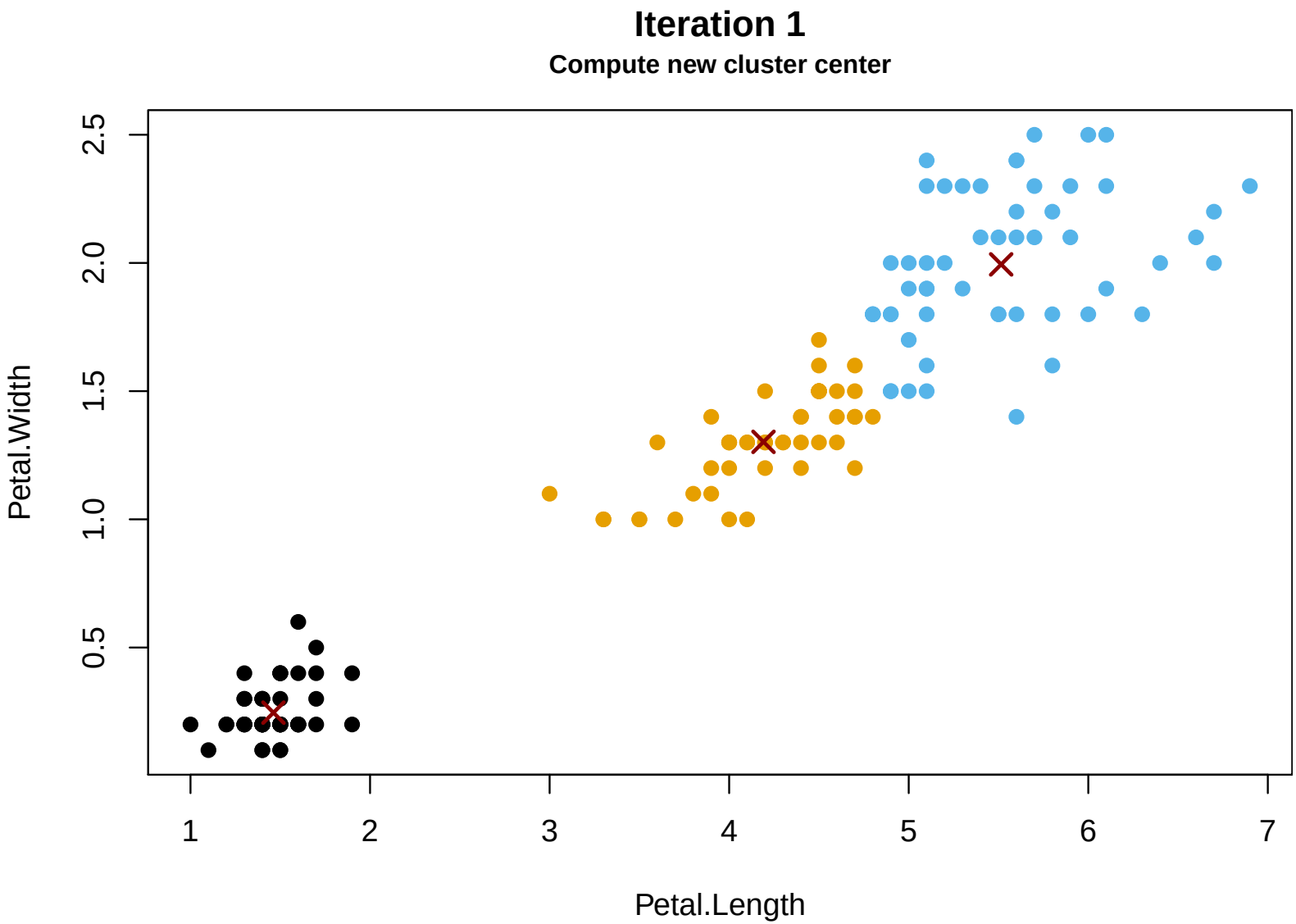
K-MEANS EXAMPLE



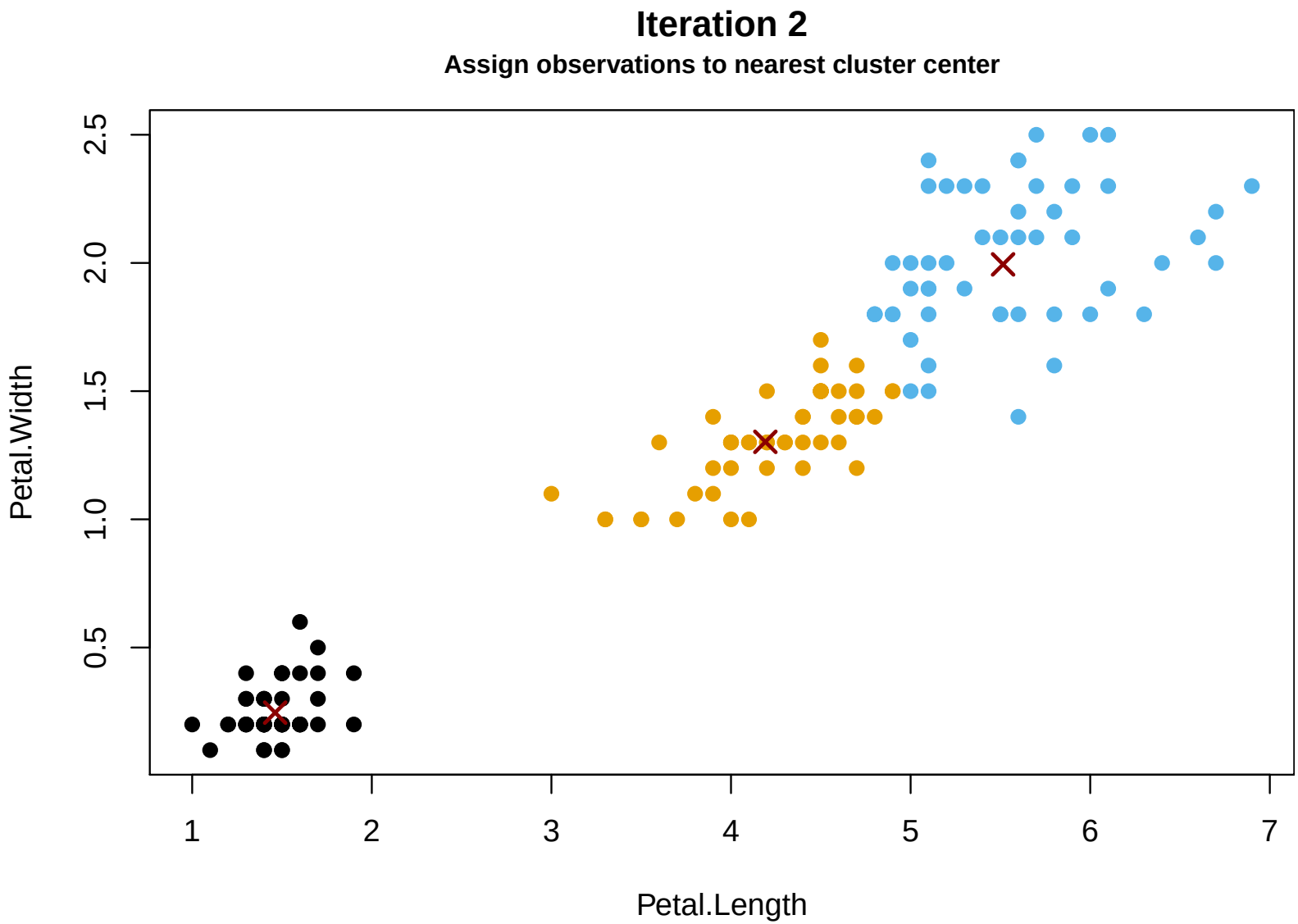
K-MEANS EXAMPLE



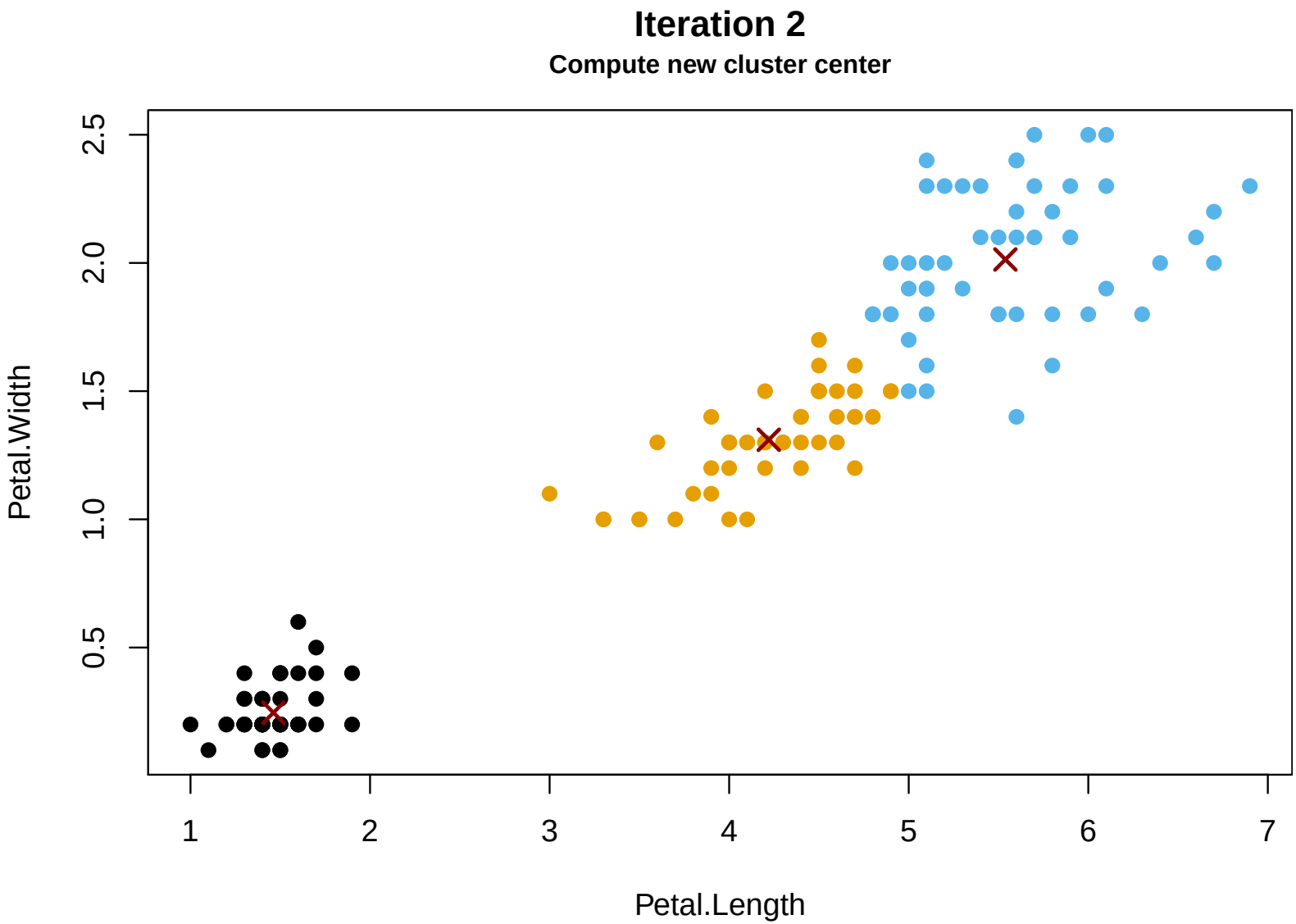
K-MEANS EXAMPLE



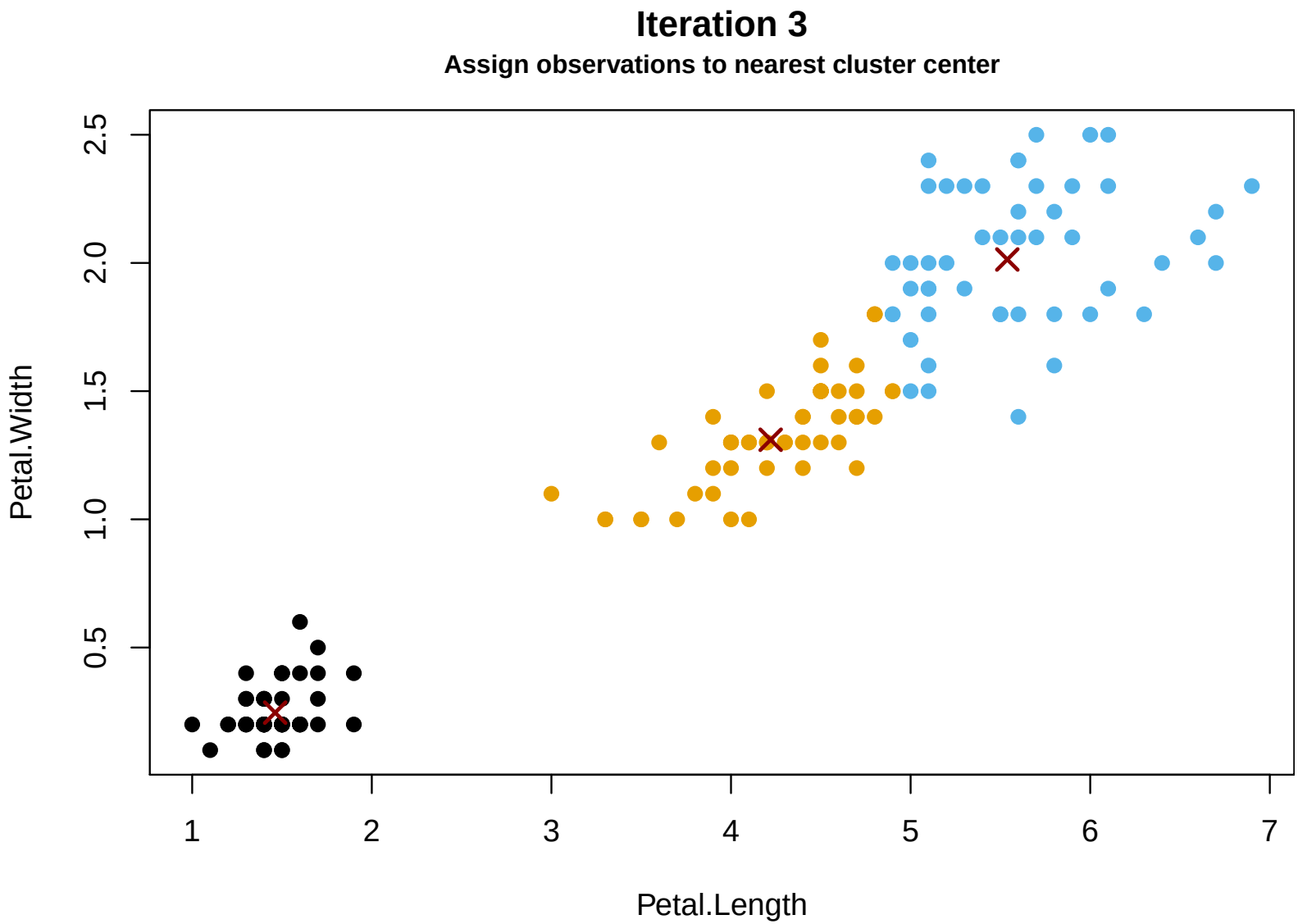
K-MEANS EXAMPLE



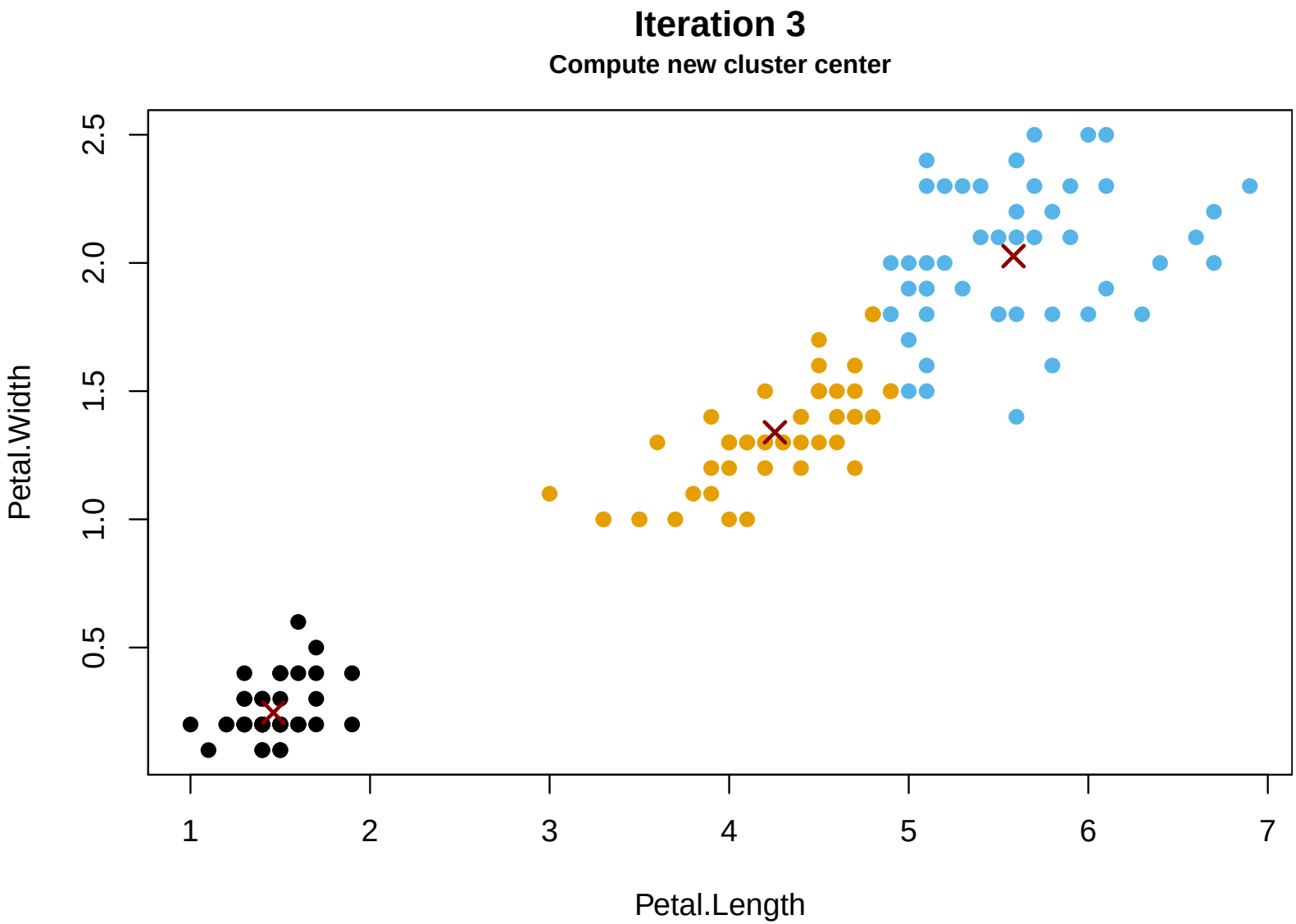
K-MEANS EXAMPLE



K-MEANS EXAMPLE



K-MEANS EXAMPLE

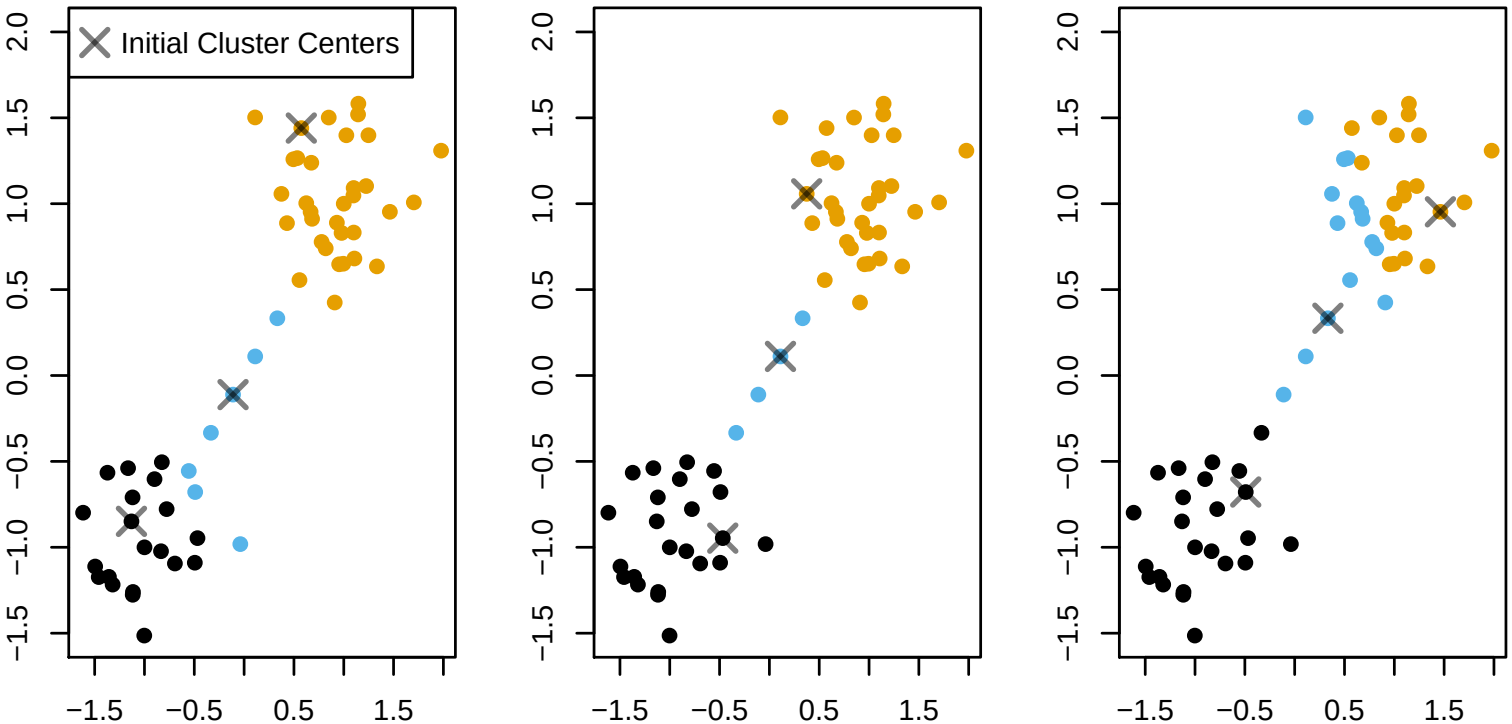


PROPERTIES OF K-MEANS

- k-means is based on computing the mean, which is sensitive to outliers and can only be computed for numerical data.
- The **within-cluster variation** is reduced in each iteration.
- The final result is typically not the best result that globally minimizes the **within-cluster variation**.
  $\rightarrow$ would only be possible after trying all possible partitions!
- k-means can be restarted multiple times. The clustering with the smallest within-cluster variation is then selected as the best solution.

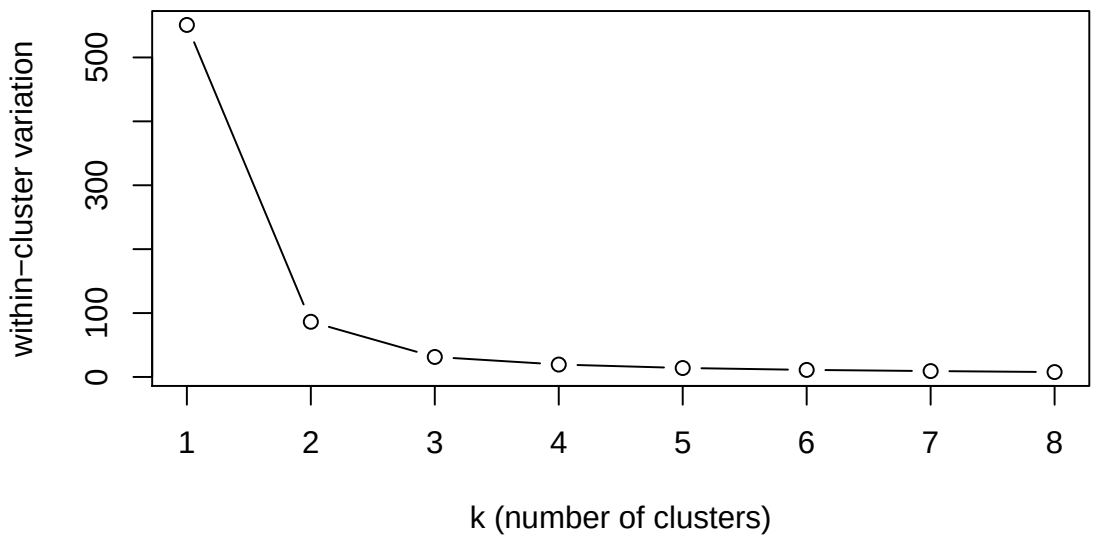
PROPERTIES OF K-MEANS

- k-means produces different clusters depending on the initial centers and always converges, e.g.:



CHOICE OF K

- Many methods exist for choosing the number of clusters k (there is no perfect solution).
- The easiest method is to apply k-means for different k and plot the **within-cluster variation** for each number of k.
- The **within-cluster variation** always decreases with increasing number of clusters.
- An **"elbow"** in the plot might indicate a useful solution.



K-MEDOIDS

- is strongly related to *k*-means and is realized by the **Partitioning Around Medoids (PAM)** algorithm.
- uses cluster medoids as representative clusters, i.e. **real data points** instead of **artificial data points** (such as the cluster centers as in *k*-means) are used.
- is less sensitive to outliers and more robust than *k*-means.
- can handle categorical features (*k*-means does not because it is based on calculating the cluster centers by taking the mean in each dimension).

THE PAM ALGORITHM

1. **Initialization:** Randomly select k data points as the medoids.
2. **Assignment:** Assign each data point $\mathbf{x}^{(i)}$ to its closest medoid m and calculate the within-cluster variation for each medoid (by summing up the distances of the current medoid m to all other data points associated to m) .
3. **Update:** Swap m and $\mathbf{x}^{(i)}$ and recompute the within-cluster variation to see if another medoid is more appropriate. Select the medoid m with the lowest within-cluster variation.
4. Repeat steps (2) and (3) until medoids do not change.

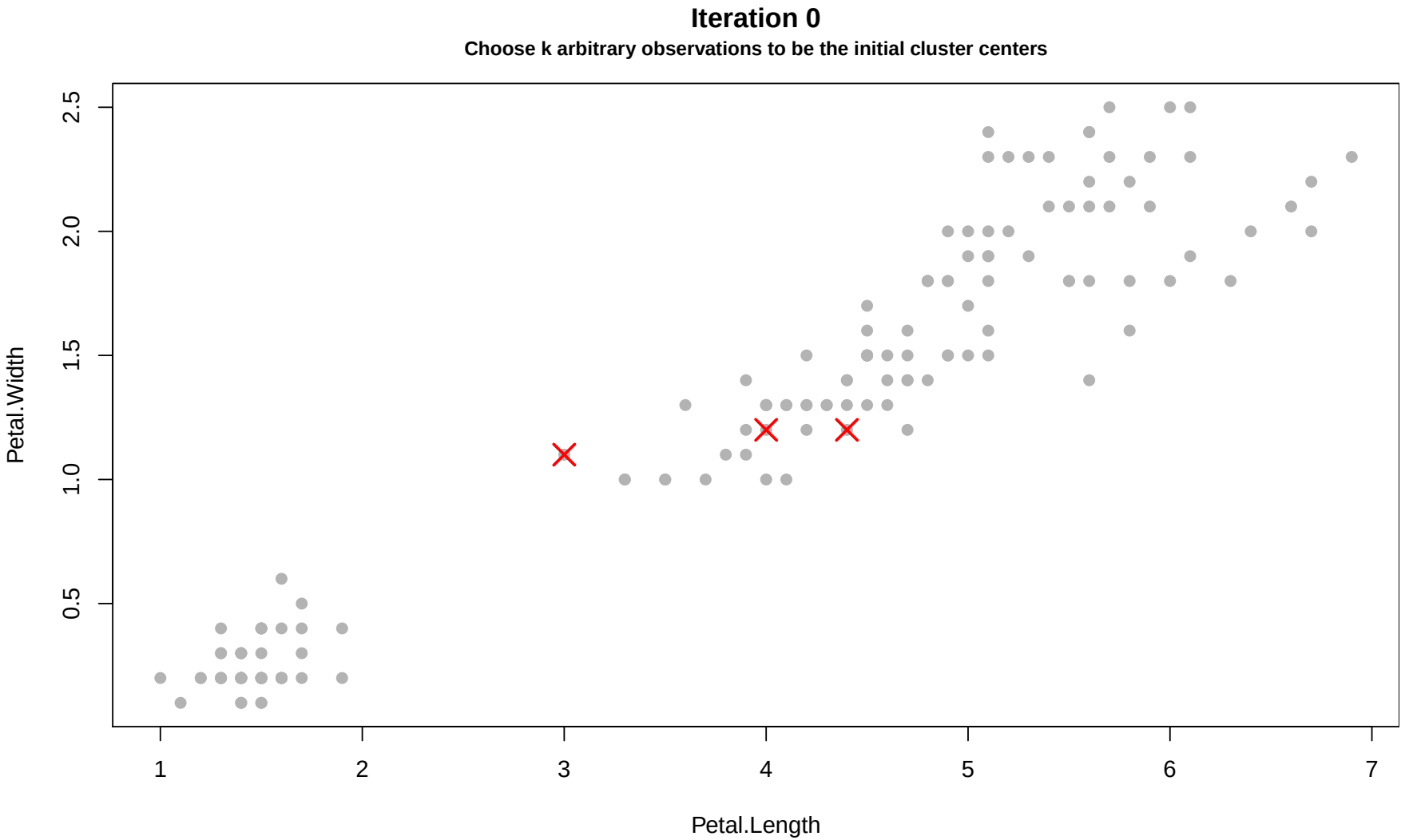
THE PAM ALGORITHM

The PAM algorithm typically uses the following two metrics to compute distances:

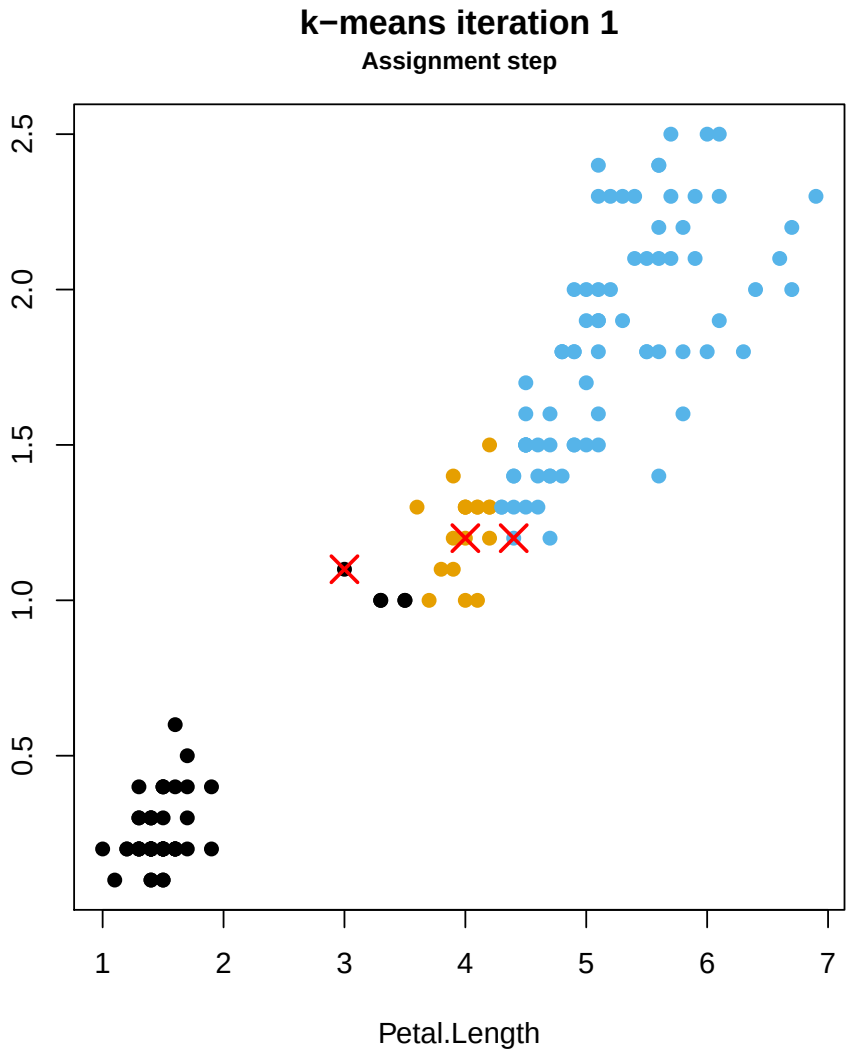
- The euclidean distance (root sum-of-squares of differences).
- The Manhattan distance (the sum of absolute distances).

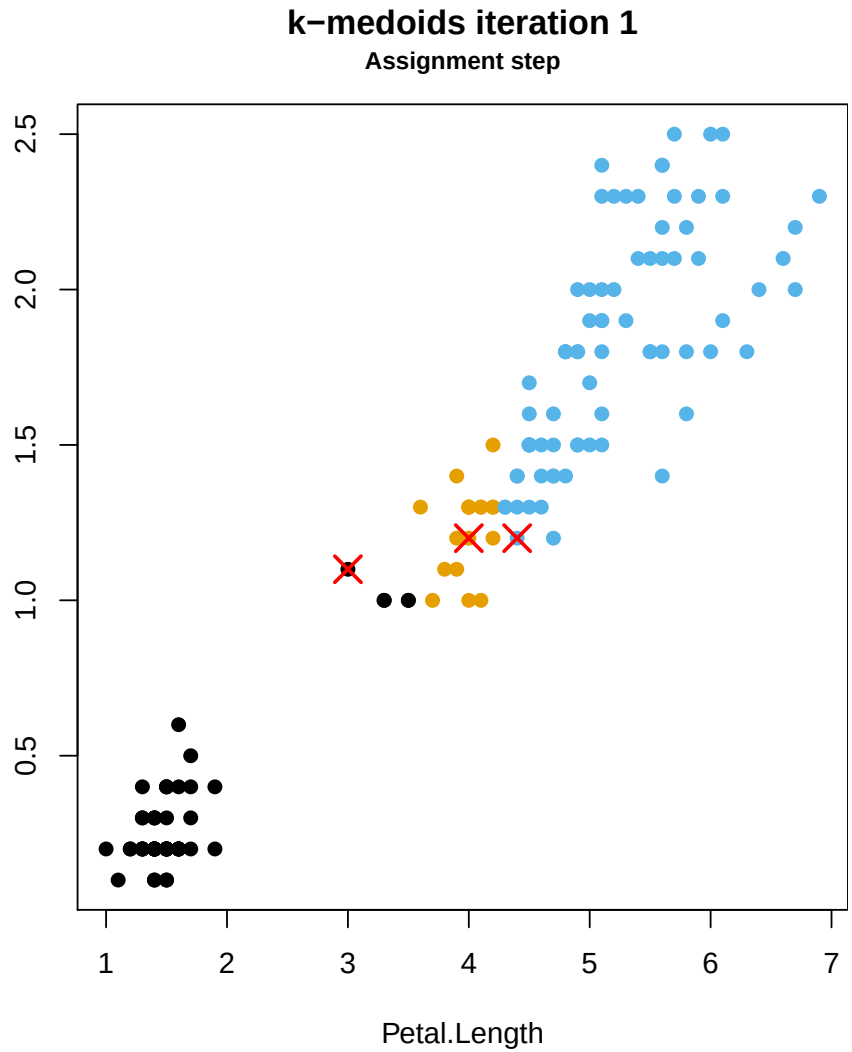
Note: The Manhattan distance should give more robust results if your data contains outliers. In all other cases, the results will be similar for both metrics.

K-MEANS VS. K-MEDOIDS

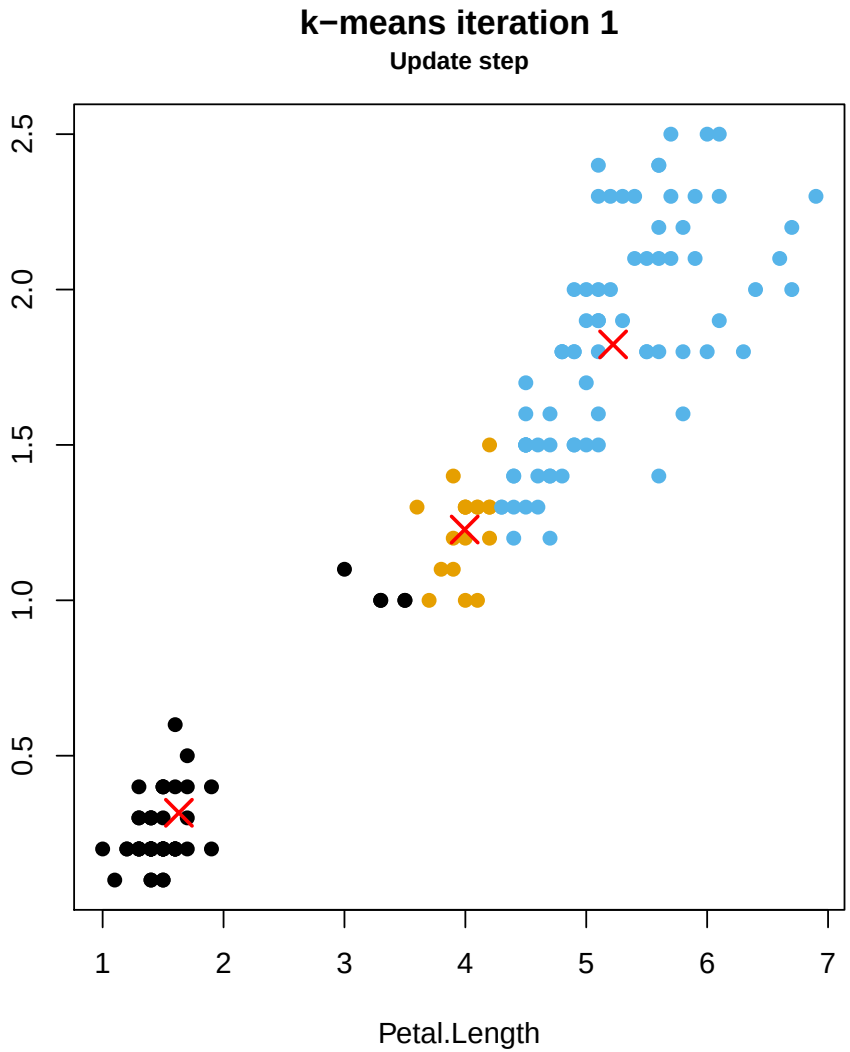


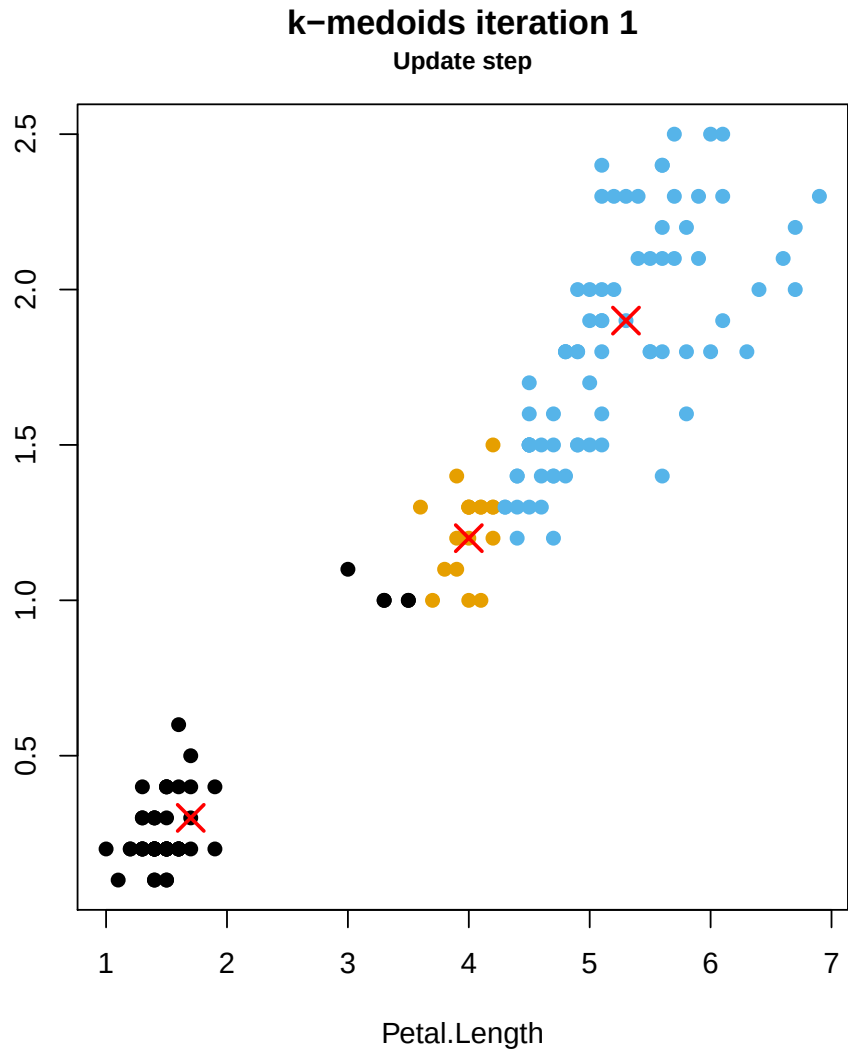
K-MEANS VS. K-MEDOIDS



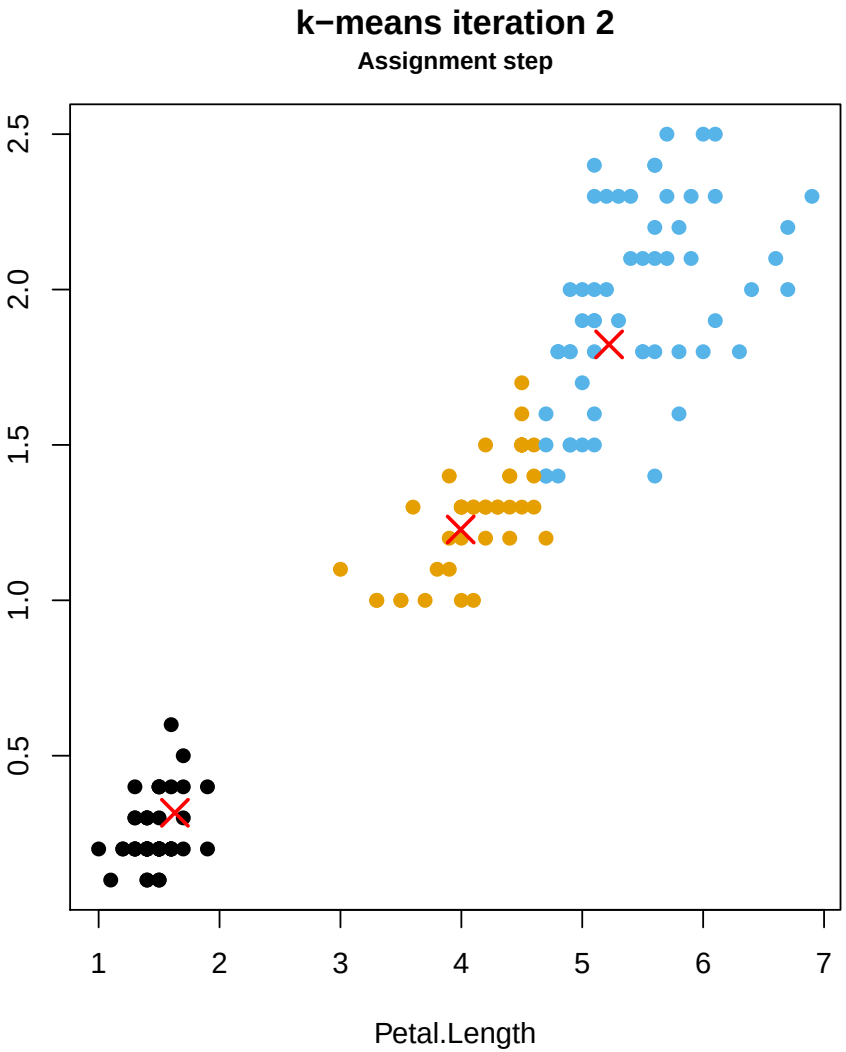


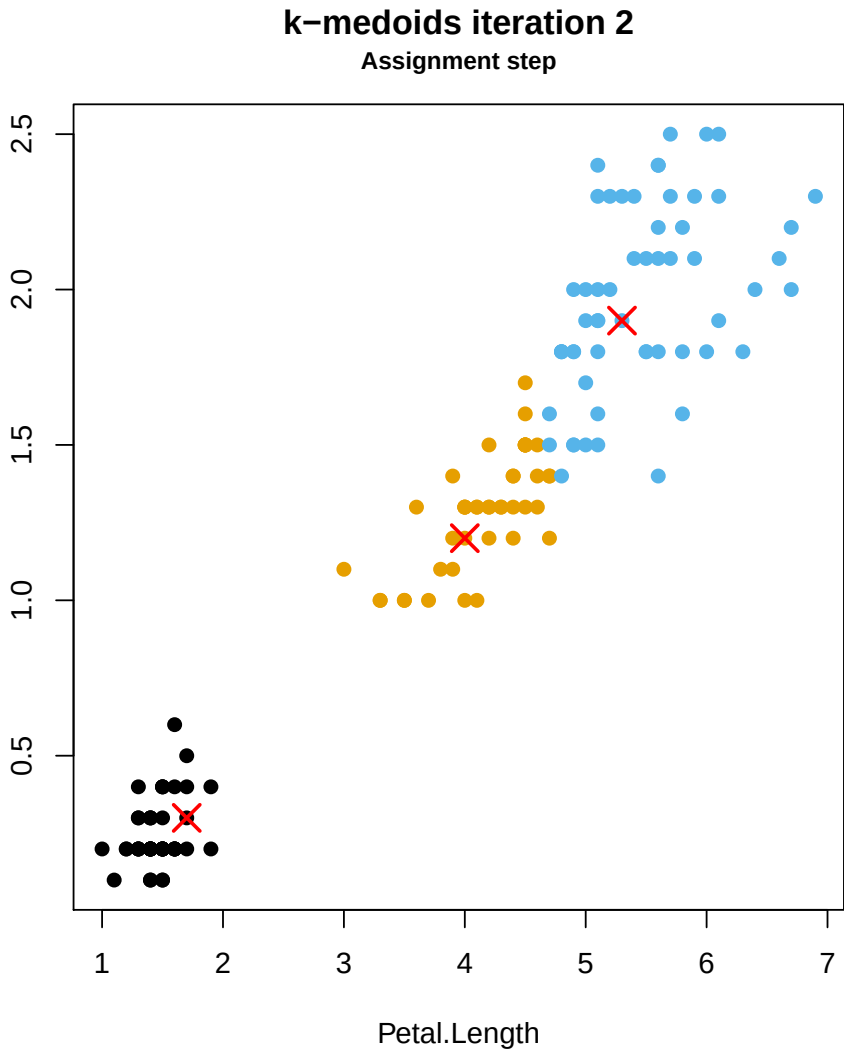
K-MEANS VS. K-MEDOIDS



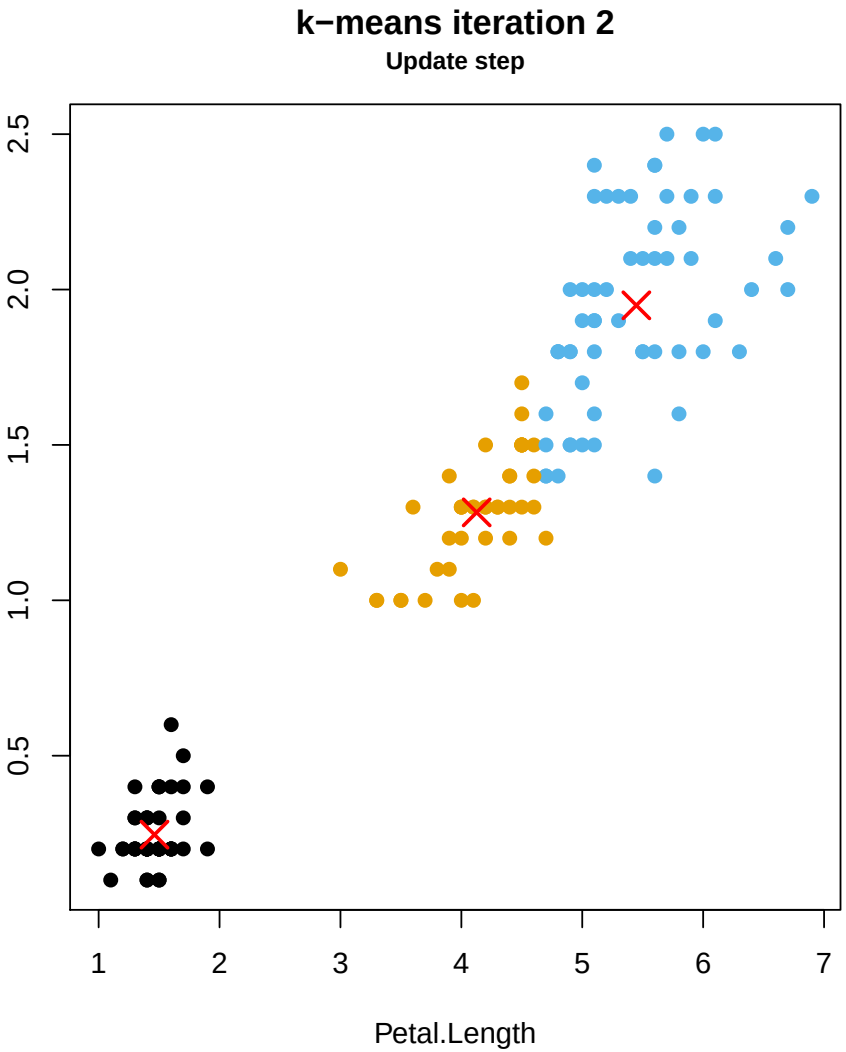


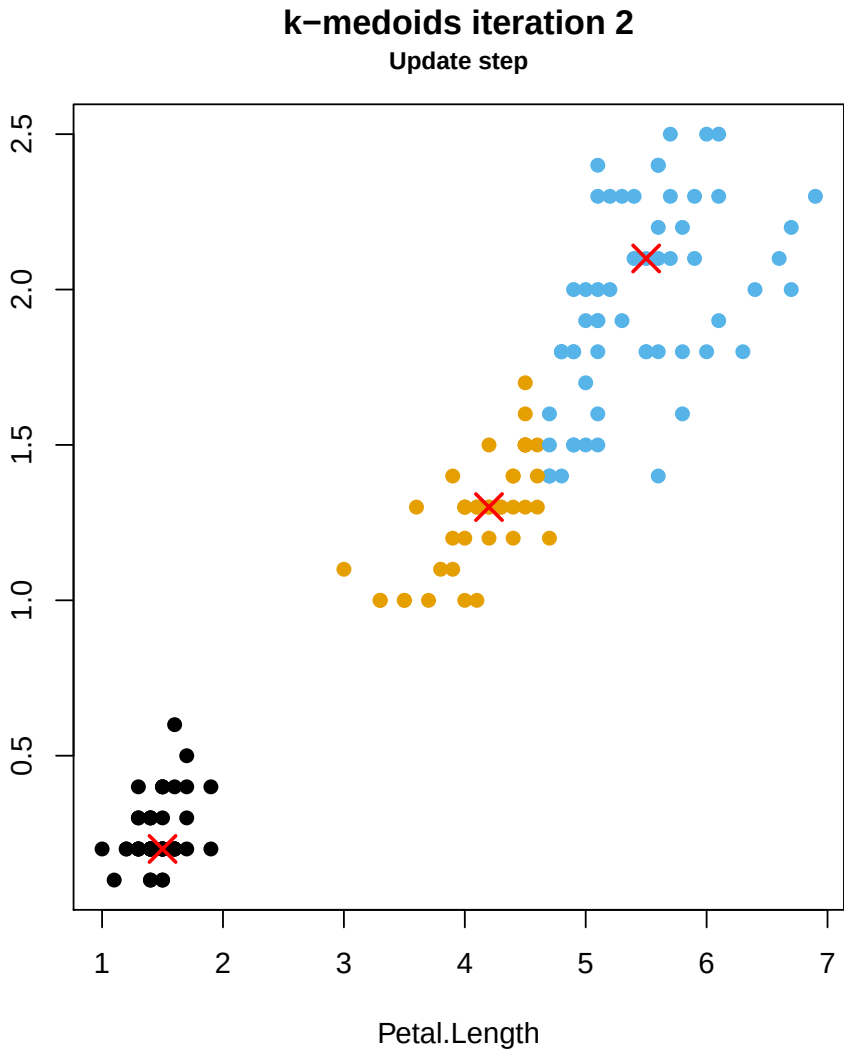
K-MEANS VS. K-MEDOIDS





K-MEANS VS. K-MEDOIDS





SUMMARY

- Minimizing the **within-cluster variation** exactly is not feasible and can be approximated by the k-means algorithm.
- k-means always converges, however, the cluster assignments strongly depend on the initial centers.
  $\rightarrow$ repeat it several times with different initial centers.
- A simple solution for choosing the number of clusters k is to plot the **within-cluster variation** for several k and look for an **"elbow"** which is a good guess for k.