A comparison of clustering algorithms

Verena Brufatto

August 10, 2021

To Elide

Contents

1 Introduction

The purpose of this exercise is to implement and compare different clustering techniques, which are aimed at partitioning the data $X \in \mathbb{R}^q$ into K homogeneous clusters. The homogeneity of the clusters is measured by a dissimilarity function which is the objective function to minimize in order to find the optimal clusters. Clustering techniques belong to the class of unsupervised learning methods, since they are based solely on the features of the data and do not take into consideration the response variable.

The clustering techniques are applied to the Boston Housing dataset, which is composed of 506 observations of real estate market data collected in Boston, Massachusetts, in 1978.

We implement two centroid-based (K-means and K-medoids) and one distribution-based (Gaussian mixture models) clustering techniques, which require the number of clusters to be specified as a hyperparameter. Furthermore, we implement an agglomerative hierarchical clustering algorithm, which recursively merges clusters into bigger clusters and does not require to choose the number of clusters *a priori*. The selected techniques are evaluated based on how well they are able to classify houses with a high market price as well as on criteria that measure cluster homogeneity.

2 Data pre-processing

Out of the 14 features of the original Boston Housing dataset, we consider only the following:

- *nox*: nitric oxides concentration (parts per 10 million)
- *rm*: average number of rooms per dwelling
- *tax*: full-value property-tax rate per USD 10,000
- *ptratio*: pupil-teacher ratio by town school district
- *lstat*: percentage of lower status of the population
- *medv*: median value of owner-occupied homes in USD 1000's (target variable)

The features have been chosen based on their correlation with the target variabile medv (Table 1).

As in the original paper (Harrison and Rubinfeld (1978)), we apply a logarithmic transformation to the variables *medv* and *lstat* and a quadratic transformation to the variables *nox* and *rm*.

Figure 1 shows the histogram and boxplot of the median value of houses. The distribution has positive skewness, a mean value of about 23 thousand USD, a median value of 21 thousand USD and is capped at 50 thousand USD. All values above 39 thousand USD are considered outliers, since they make up for only 7% of all observed house prices. For the purpose of this analysis, we label house prices above the 90th percentile, which corresponds to 35 thousand USD, as “expensive” (Table 2).

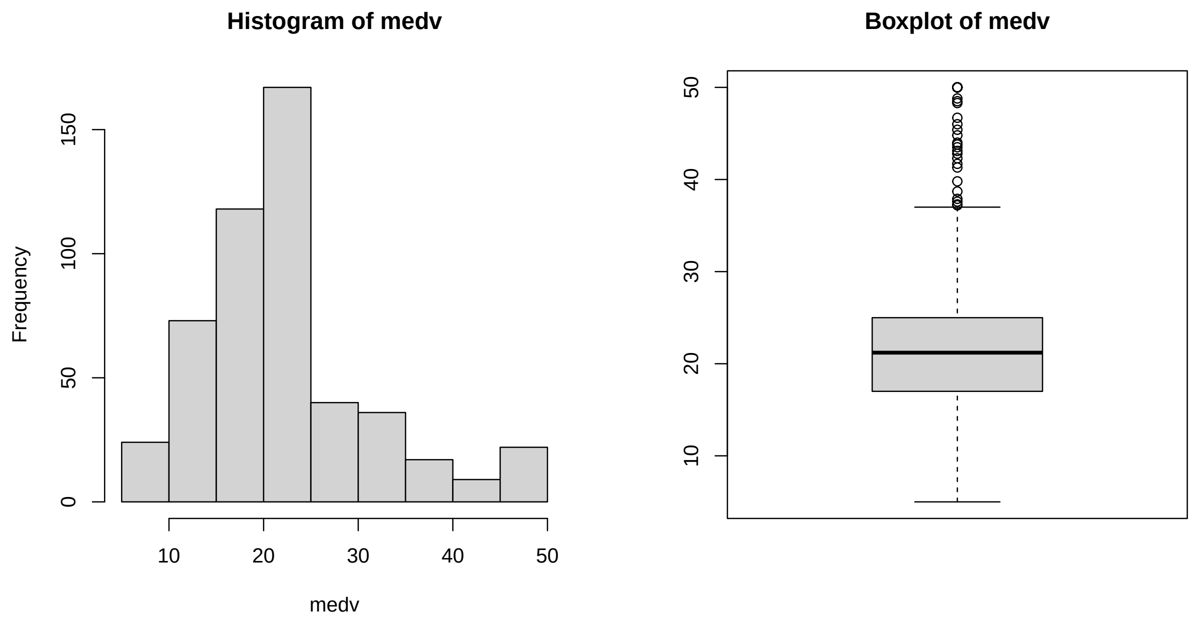
Table 1: Feature correlation

	medv	lstat	ptratio	tax	rm	nox
medv	1.00	-0.74	-0.51	-0.47	0.70	-0.43
lstat	-0.74	1.00	0.37	0.54	-0.61	0.59
ptratio	-0.51	0.37	1.00	0.46	-0.36	0.19
tax	-0.47	0.54	0.46	1.00	-0.29	0.67
rm	0.70	-0.61	-0.36	-0.29	1.00	-0.30
nox	-0.43	0.59	0.19	0.67	-0.30	1.00

Table 2: Descriptive statistics

n	missing	distinct	Mean	.05	.10	.25	.50	.75	.90	.95
506	0	229	22.53	10.20	12.75	17.02	21.20	25.00	34.80	43.40

Figure 1: Median house value



3 K-means clustering

K-means is a centroid-based clustering algorithm that partitions the data $x_i \in X$, $i = 1, ..., n = 506$ into K disjoint clusters $(C_1, ..., C_K)$

$$C_k = \{x \in \mathbb{R}^q; C_K(x) = k\}, \forall k \in K$$

so that

$$\bigcup_{k=1}^{K} C_k = \mathbb{R}^q \qquad \text{and} \qquad C_k \cap C_l = \emptyset, \forall k \neq l$$

The number of clusters K is a hyperparameter that must be chosen *a priori*, while the clusters are built so that the dissimilarity of the elements belonging to each cluster C_k is minimal.

The dissimilarity function chosen is the squared Euclidean distance on $\mathbb{R}^q$

$$d(x, x') = ||x' - x||_2^2 = \sum_{j=1}^{q} (x'_j - x_j)^2$$

The clusters are obtained by minimizing the total within cluster dissimilarity (TWCD)

$$\underset{(C_1,...,C_K)}{argmin} \sum_{k=1}^{K} \sum_{x_i \in C_k \cap X} d(\mu_k, x_i) = \underset{(C_1,...,C_K)}{argmin} \sum_{k=1}^{K} \sum_{x_i \in C_k \cap X} ||\mu_k - x_i||_2^2$$

where $X = \{x_1, ..., x_n\}$ and μ_k is the sample mean over a single cluster C_k, which is also called cluster center or centroid

$$\mu_k = \frac{1}{|\{x_i \in C_k \cap X\}|} \sum_{x_i \in C_k \cap X} x_i \in \mathbb{R}^q$$

For a single cluster C_k, the sample mean μ_k minimizes the within-cluster dissimilarity $D(C_k, \mu)$

$$\mu_k = \underset{\mu \in \mathbb{R}^q}{argmin} \quad D(C_k, \mu) = \underset{\mu \in \mathbb{R}^q}{argmin} \sum_{x_i \in C_k \cap X} ||\mu - x_i||_2^2$$

where

$$D(C_k, \mu_k) = \sum_{x_i \in C_k \cap X} ||\mu_k - x_i||_2^2$$

Hence, the total within-cluster dissimilarity is minimized by computing the optimal clusters $(C_1, ..., C_K)$

$$\underset{(C_1,...,C_K)}{argmin} \sum_{k=1}^{K} D(C_k, \mu_k)$$

Algorithm 1 performs K-means clustering for features $X = \{x_1, ..., x_n\}$ and converges to a local minimum. Convergence is ensured due to the fact that each iteration reduces the total within-cluster dissimilarity.

Step 2a of the algorithm computes the optimal sample means μ_k^{t-1} with respect to the dissimilarity measure, while step 2b updates the centroids with respect to the new clusters C_K^t.

The objective function, i.e. the within-cluster dissimilarity, decreases with each iteration while having a lower bound of zero, hence ensuring convergence. In order to avoid local minima, the initial classifier C_K^0 can be randomly restarted in the first step of the algorithm.

Algorithm 1 K-Means clustering

```
1. Choose an initial clustering classifier $C_K^0 : X \to K$ with sample means $\mu_k^0$, $k \in K$.
2. Repeat for $t \geq 1$ until there are no further changes:
    (a) given the current sample means $\mu_k^{t-1}$ choose the classifier $C_K^t : X \to K$ so
        that for each $x_i \in X$
            $C_K^t(x_i) = \underset{k \in K}{argmin} ||\mu_k^{t-1} - x_i||_2^2$
    (b) calculate the sample means $\mu_k^t$ on $C_k^t$.
```

The number of clusters K is a hyperparameter that must be chosen *a priori*. To ensure that the total within-cluster dissimilarity is decreasing in K, one can start by forming $K = 2$ clusters C_k with sample means μ_k for $k = 1, 2$. For $K = 3$ the sample means μ_1 and μ_2 can be used as initial values for the algorithm, while $\mu_3 \in \mathbb{R}^q$ can be randomly initialized. The process can then be repeated for the desired number of clusters K.

The following chunk of code performs k-means using the `kmeans` function from the R package `stats`.

```r
features = dataset[, -which(colnames(dataset) == "medv")] # exclude target variable
X = scale(features) # normalize features

k_max = 10 # set maximum number of clusters
dissimilarity = matrix(0, nrow = k_max, ncol = 1) # within-cluster dissimilarity
clusters = matrix(1, ncol = k_max, nrow = nrow(X)) # clusters

means = colMeans(X) # feature means (zero since normalized)
dissimilarity[1] = sum(colSums(X^2)) # dissimilarity for 1 cluster

set.seed(123)
for(k in 2:k_max){
  # form clusters for k = 2, ..., k_max
  if(k == 2){
    # kmeans for 2 clusters
    k_mean = kmeans(X, k)

  } else {
    # kmeans with initial centroid values
    k_mean = kmeans(X, k_centers)
  }

  dissimilarity[k] = sum(k_mean$withins)
  clusters[, k] = k_mean$cluster
  k_centers = matrix(0, nrow = k+1, ncol = ncol(X)) # initial centroid values
  k_centers[1:k, ] = k_mean$centers # centroids for k
  k_centers[k+1, ] = means # feature means for k+1
}
```

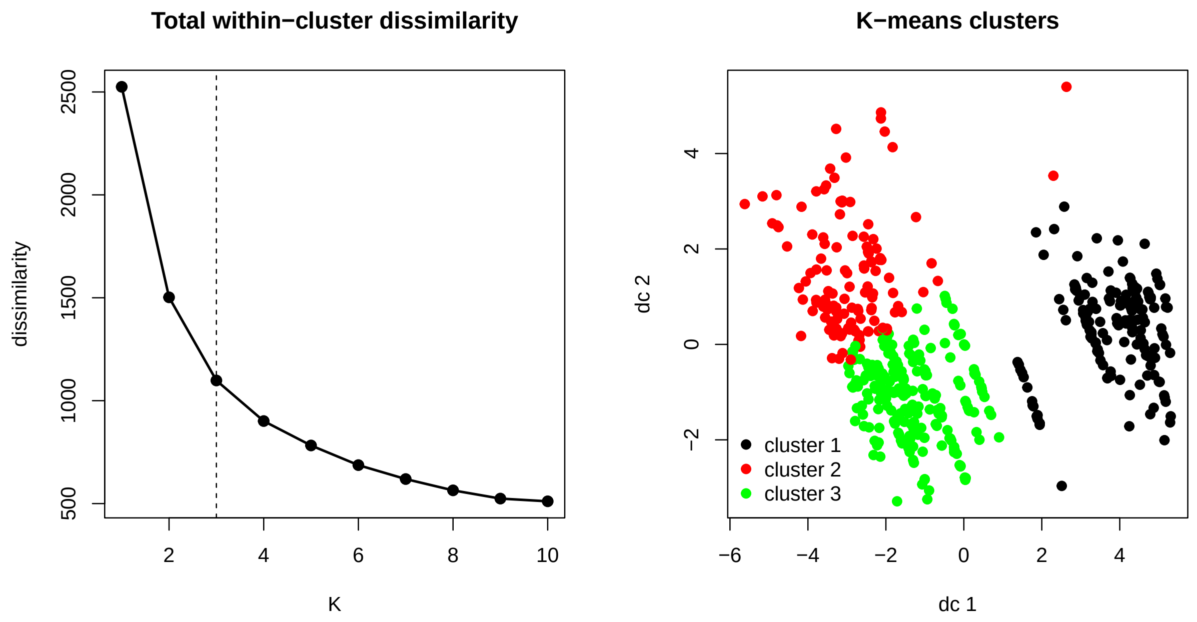
The k-means algorithm is run for $K = 2, ..., 10$ clusters and, at each iteration, the previous cluster centroids $\mu_k, k \in K$, are used as initial values for the centroids in the next iteration, so that the total within-cluster dissimilarity decreases in K. The resulting plot (Figure 2, lhs) can be used to select the optimal number of clusters K. In this case, we choose $K = 3$ based on the elbow method.

Table 3: K-means

	cluster 1	cluster 2	cluster 3
number of houses	166	121	219
expensive houses	4	46	1
in %	8%	90%	2%

Table 3 shows the results of k-means with respect the whole sample and to houses with a high market value. Most of the houses belong to cluster 3 (43%) and 90% of the houses with a high market price belong to cluster 2.

Figure 2: K-means



Next, we perform principal component analysis (PCA) with the R function `prcomp` and extract the first two principal components, which explain about 75% of the total variance in the data.

Let $q \leq n$ be the rank of the feature matrix X, so that there are q linearly independent samples $x_i \in X$ that span the whole space $\mathbb{R}^q$. PCA determines an orthonormal basis $z_1, ..., z_p \in \mathbb{R}^p$, with $p \leq q$, so that the q-dimensional representation $x_i \in X$ may be replaced by a p-dimensional representation while preserving as much of the original variability as possibile.

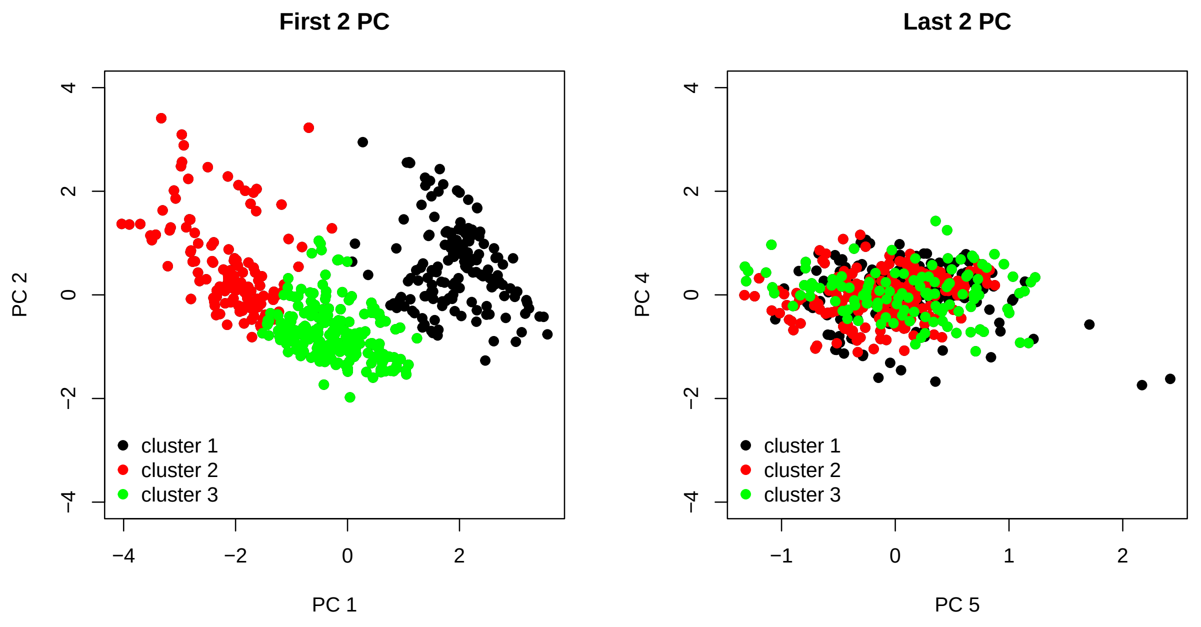
```
PCA = prcomp(X) # principal component analysis
dati_pca = cbind( X %*% PCA$rotation[, 1], # 1st and 2nd principal component
                  X %*% PCA$rotation[, 2] )

summary(PCA)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5
## Standard deviation     1.6620 0.9723 0.8811 0.52636 0.48881
## Proportion of Variance 0.5525 0.1891 0.1553 0.05541 0.04779
## Cumulative Proportion  0.5525 0.7415 0.8968 0.95221 1.00000
```

Figure 3 shows the k-means clusters with respect to the first two and the last two principal components. Considering the first two principal components (lhs), we obtain clusters that are well defined, which indicates that k-means clustering is essentially based on the two principal components that explain most of the variability in the data. In fact, if we consider the two principal components associated with the smallest explained variance, we observe significant overlap among cluster members (rhs).

Figure 3: K-means vs PCA



4 K-medoids clustering

K-medoids clustering is a centroid-based clustering algorithm similar to k-means clustering. The main difference between the two is that k-medoids uses data points (the *medoids*) x_i as cluster centers, whereas k-means uses the sample mean (the *centroid*) of the data μ_k. Furthermore, the k-medoids algorithm supports other dissimilarity functions besides the squared Euclidean distance, which may be more robust (e.g. with respect to ouliers).

The objective function to be minimized is the following

$$\underset{(c_1,\ldots,c_k)\subset X}{argmin} \sum_{k=1}^{K} \sum_{x_i \in C_k \cap X} d(c_k, x_i)$$

where $c_k \in X$ are the medoids belonging to the dataset, $d(.,.)$ is a dissimilarity function on $\mathbb{R}^q$ and the clusters are given by

$$C_k = \{x \in X; \quad d(c_k, x) < d(c_l, x), \quad \forall l \neq k\}$$

Since the medoids belong to the data set, the dissimilarities $d(x_i, x_l), i \neq l$ need to be calculated only once. The resulting dissimilarity matrix can then be directly provided to the algorithm.

In this example, we choose the Manhattan distance as the dissimilarity function

$$d(x, x') = \sum_{j=1}^{q} |(x'_j - x_j)|$$

Algorithm 3 performs K-medoids clustering for features $X = \{x_1, ..., x_n\}$ and converges to a local minimum (Kaufman-Rousseeuw (1987)).

Algorithm 2 K-Medoids clustering

1. Choose initial medoids $c_1, ..., c_k \in X$ and assign each data point $x_i \in X$ to its closest medoid. Then, calculate the total within-cluster dissimilarity

$$TWDC = \sum_{k=1}^{K} \sum_{x_i \in C_k \cap X} d(c_k, x_i)$$

2. Repeat there is no further decrease in TWDC. For each c_k and for each x_i:
   (a) set x_i as the new medoid c_k and allocate each data point to the respective cluster
   (b) calculate the new TWDC
   (c) if TWDC decreases accept x_i as the new medoid, otherwise reject the swap.

The following chunk of code performs k-medoids clustering with $K = 3$ using the R function `pam` from the package `cluster`. Since we chose the Manhattan distance as a dissimilarity function, we do not provide a dissimilarity matrix to the algorithm (`diss = F`), so that the features X are considered only as observations. The parameter `pamonce = F` corresponds to the original algorithm by Kaufman-Rousseeuw (1987).

```r
set.seed(123)
k_medoid = pam(X, k = 3, metric = "manhattan", diss = F, pamonce = F)
```

Figure 4 shows the dissimilarity function with respect to K, computed with the function `fviz_nbclust` of the package `factoextra` (lhs), and the clusters produces by the k-medoids algorithm (rhs).

Most of the houses belong to cluster 1 (36%) and 88% of the houses with a high market price belong to cluster 1. Hence, the classification error for expensive houses is slighly larger for the k-medoids algorithm compared to k-means (Table 4).

Figure 5 shows a comparison of the k-means and k-medoids results with respect to the first two principal components. We observe that, except for the different cluster naming, the two methods yield similar results in terms of clusters. Furthermore, we observe that the cluster centers are closer for k-medoids than for k-means, due to the fact that the Manhattan distance is more flexible with respect to outliers than the Euclidean distance. Lastly, k-medoids clustering appears to produce a higher number of classification errors than k-means, especially with respect to cluster 2.

Figure 4: K-medoids

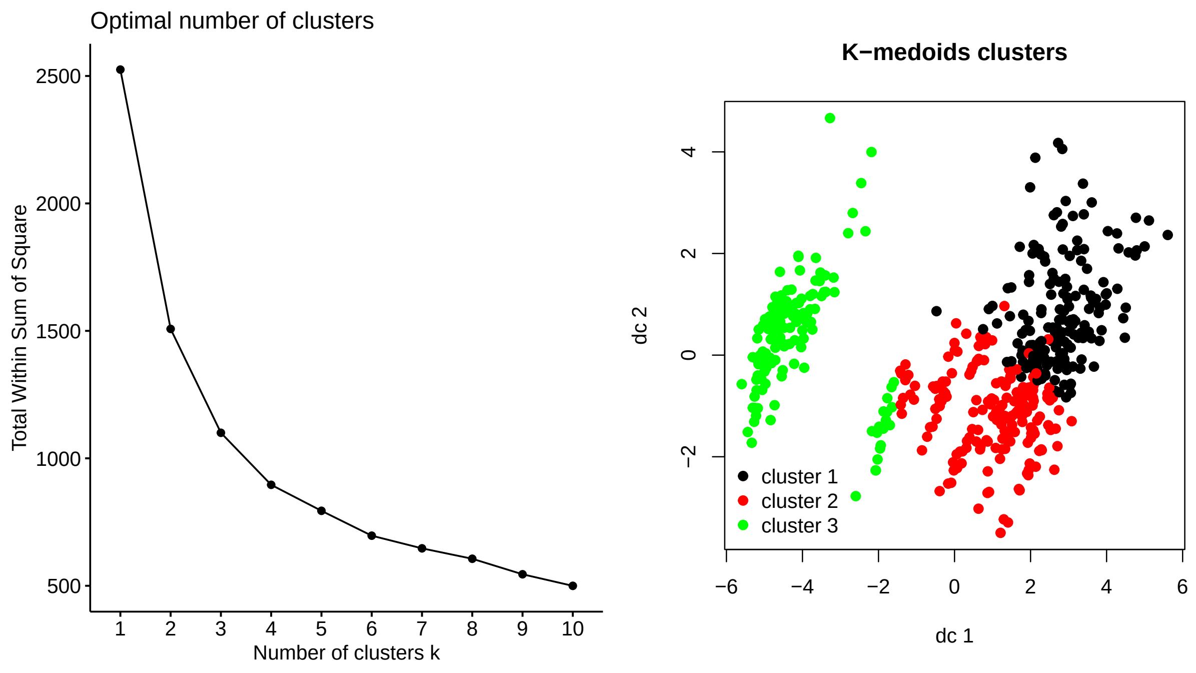
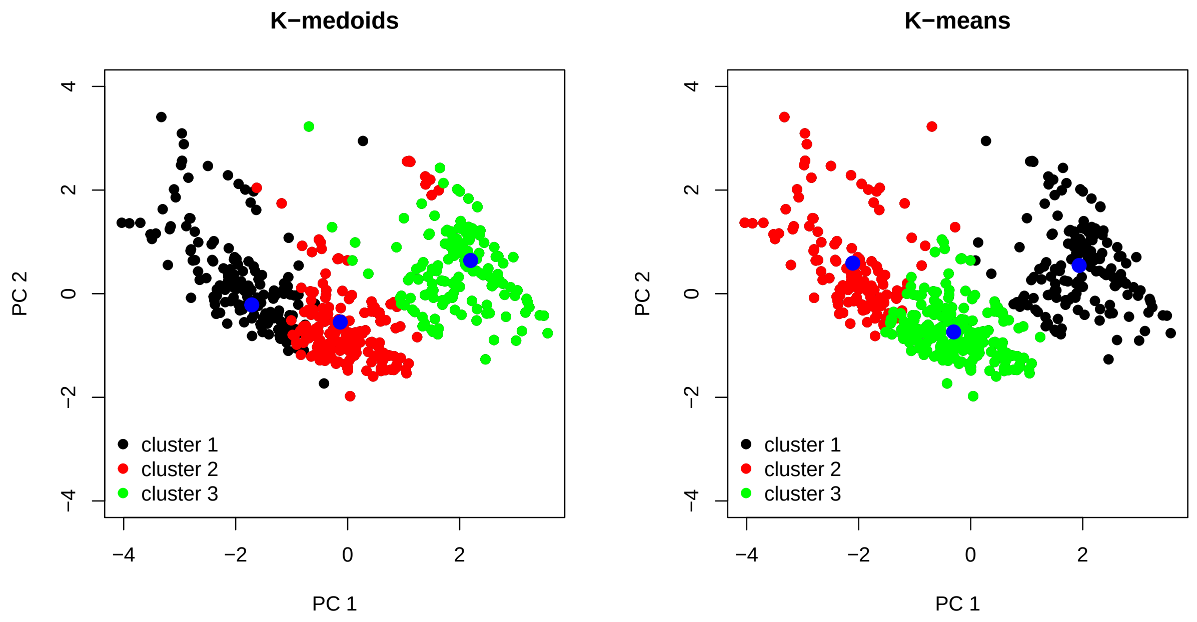


Table 4: K-medoids

	cluster 1	cluster 2	cluster 3
number of houses	170	180	156
expensive houses	45	1	5
in %	88%	2%	10%

Figure 5: K-medoids and K-means vs PCA



5 Clustering with Gaussian mixture models

Gaussian mixture models (GMM) are distribution-based clustering models that rely on a probabilistic assumption about the data generating process. They can be seen as a probabilistic variant of k-means, which assigns samples to clusters based on probability density rather than distance.

Given the hyperparameter $K \in \mathbb{N}$, we assume that the features $x_i \in X$ follow a multivariate GMM with parameter $\theta = (\mu_k, \Sigma_k, p_k)$, $k \in K$, i.e. they are iid realizations from the density of a weighted sum (mixture) of normal distributions

$$f(x) = \sum_{k=1}^{K} \frac{1}{(2\pi|\Sigma_k|)^{q/2}} exp\{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\} p_k$$

where $\mu_k \in \mathbb{R}^q$ are the mean vectors, $\Sigma_k \in \mathbb{R}^{q \times q}$ are the covariance matrices and weights $p = (p_1, ..., p_K)$ so that $p_k \geq 0$ and $\sum_{k=1}^{K} p_k = 1$. In this framework, the cluster centers are given by the Gaussian mean vectors μ_k.

In order to simplfy the optimization problem, we introduce a latent variable $Z = (Z^1, ..., Z^K) \in S_K$ which characterizes the Gaussian distribution from which a particular observation x has been sampled. Hence, Z is a K-dimensional vector that takes on values between 0 and 1 such that $\sum_{k=1}^{K} Z_k = 1$.

The multivariate GMM can be rewritten as

$$f(x) = \sum_{z \in S_K} f(x, z)$$

so that $(x_i, z_i) \in \mathbb{R}^q \times S_k$, $i = 1, ..., n$ is iid and has joint density

$$f(x,z) = f(x|z)p(z) = \sum_{k=1}^{K} z^k \frac{1}{(2\pi|\Sigma_k|)^{q/2}} exp\{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\}p_k$$

where $p_k = \mathbb{P}[Z^k = 1] > 0$, $k \in K$ is the probability that a sample is generated by the k-th Gaussian, i.e. belongs to the k-th cluster.

The parameter θ can then be estimated by maximum likelihood. The log-likelihood function for (x_i, z_i) is much more tractable than for x_i and is given by

$$l_{(x_i,z_i)}(\theta) = \sum_{i=1}^{n}\sum_{k=1}^{K} z_i^k \, log \, f(x_i|\mu_k, \Sigma_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} z_i^k \, log(p_k)$$

The maximum likelihood estimators (MLE) for the parameters are given by

$$\hat{\mu}_k = \frac{\sum_{i=1}^n z_i^k x_i}{\sum_{i=1}^n z_i^k}$$

$$\hat{\Sigma}_k = \frac{\sum_{i=1}^n z_i^k (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{\sum_{i=1}^n z_i^k}$$

and

$$\hat{p}_k = \frac{1}{n}\sum_{i=1}^{n} z_i^k$$

Since the latent variables Z_i cannot be observed, the GMM model is estimated with the expectation-maximization (EM) algorithm, which consists of two steps:

1. Expectation step: estimate Z_i from x_i and $\hat{\theta}$. Since Z_i is not observable, it is estimated by its posterior expectation given observation x

$$\hat{Z}^k(\theta|x) = \mathbb{E}[Z^k|x] = p_k(\theta|x) = \mathbb{P}[Z^k = 1|x]$$

2. Maximization step: estimate θ from $(x_i, \hat{Z}_i)$ by MLE.

The expectation step in the EM algorithm is equivalent to the first step of the k-means algorithm, in that we re-assess the previously formed clusters using the estimated cluster centers. Instead of assigning each data point to the best-matching cluster, we compute the posterior expectation. In the maximization step, the cluster centers μ_k, along with the other parameters, are computed based on the new clusters.

Algorithm 3 Expectation-Maximization

```
1. Choose an initial parameter $\theta^0 = (\mu_k^0, \Sigma_k^0, p_k^0), k \in K$
2. Repeat for $t \geq 1$:
    (a) Expectation step: given $\theta^{t-1}$, estimate $Z_i$, $i = 1, ..., n$
            $\hat{Z}_i^t = (p_1(\theta^{t-1}|x_i), ..., p_K(\theta^{t-1}|x_i))$
    (b) Maximization step: calculate the MLE for $\theta^t$ based on the observations $(x_i, \hat{Z}_i^t)$
            $\hat{\mu}_k^t = \frac{\sum_{i=1}^n p_k(\theta^{t-1}|x_i)x_i}{\sum_{i=1}^n p_k(\theta^{t-1}|x_i)}$
            $\hat{\Sigma}_k = \frac{\sum_{i=1}^n p_k(\theta^{t-1}|x_i)(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{\sum_{i=1}^n p_k(\theta^{t-1}|x_i)}$
            $\hat{p}_k = \frac{1}{n}\sum_{i=1}^n p_k(\theta^{t-1}|x_i)$
```

The following code performs GMM clustering using the function `Mclust` from the package `mclust` for $K = 3$ clusters. In this example, we estimate a GMM with diagonal covariance matrices Σ_k via the argument `modelNames = "EEI"`. The covariance matrix can be rewritten as

$$\Sigma_k = \lambda_k D_k A_k D_k^T$$

where λ_k is a scalar, D_k is an orthogonal matrix of eigenvectors and A_k is a diagonal matrix. Thus, EEI means "equal volumes" $\lambda_k = \lambda$, "equal shapes" $A_k = A$ and the identity matrix as the orientation $D_k = \mathbb{I}$.

```
set.seed(123)
k_gmm = Mclust(X, G = K, modelNames = "EEI")
```

Figure 6 shows the clusters obtained with GMM and with respect to the first two principal components. Relative to k-means, GMM clustering appears to have a higher number of classification errors with respect to cluster 2. The majority of samples belong to cluster 2 (51%), while 88% of expensive houses belong to cluster 1.

Figure 6: GMM

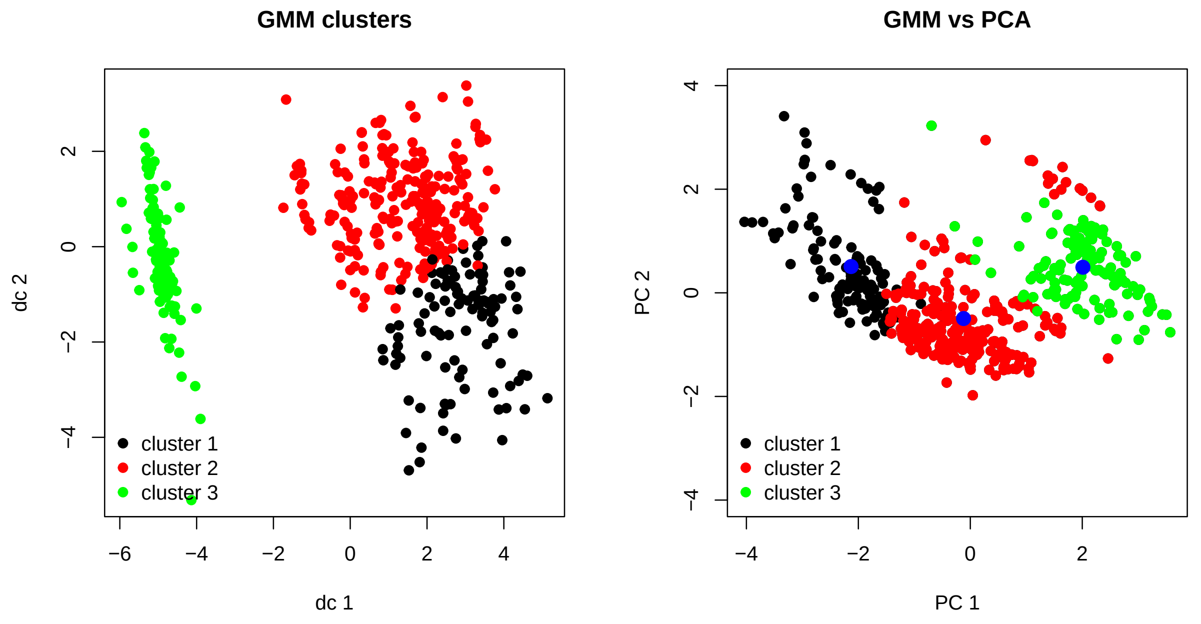


Table 5: GMM

	cluster 1	cluster 2	cluster 3
number of houses	109	260	137
expensive houses	45	1	5
in %	88%	2%	10%

6 Hierarchical clustering

Hierarchical clustering algorithms do not require to specify the number of clusters K *a priori*, but only to choose an appropriate dissimilarity function for the optimization problem. Hierarchical clustering techniques can be divided into agglomerative (bottom-up) and divisive (bottom-down). The latter consists of initializing the entire dataset as single cluster and recursively splitting each parent cluster into two daughter clusters. Agglomerative clustering algorithms instead initialize each sample as a cluster (singleton) and recursively merge pairs of clusters based on their similarity.

The dissimilarity between clusters can be measured by one of the following types of linkage:

1. **Single linkage** clustering considers the distance between the closest samples of two clusters C_k and C_l, $k \neq l$ as a measure of within-cluster dissimilarity

   $$\Delta(C_k, C_l) = \underset{x_i \in C_k, x_j \in C_l}{min} d(x_i, x_j)$$

   where $d(.,.)$ is a generic dissimilarity function such as the Euclidean distance.

2. **Complete linkage** clustering considers the most dissimilar pair of samples

   $$\Delta(C_k, C_l) = \underset{x_i \in C_k, x_j \in C_l}{max} d(x_i, x_j)$$

3. **Group average** clustering uses the average dissimilarity between clusters

   $$\Delta(C_k, C_l) = \frac{1}{|C_k||C_l|} \sum_{x_i \in C_k} \sum_{x_j \in C_l} d(x_i, x_j)$$

   where $|C_k|$ is the number of samples in cluster C_k.

In this example, we consider an agglomerative clustering algorithm based on complete linkage and use the Euclidean distance as a dissimilarity measure.

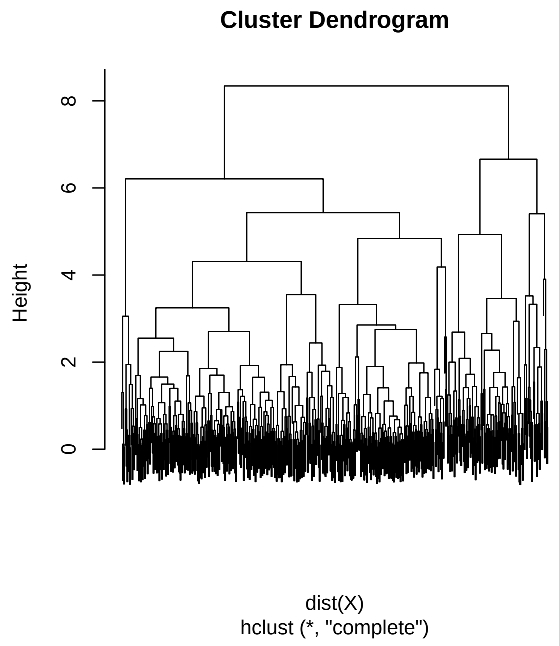
The code chunk below performs agglomerative hierarchical clustering with the function `hclust`, using complete linkage. The function `cutree` is used to cut the dendogram so as to produce $K = 3$ clusters. Figure 7 shows the cluster dendogram and the resulting clusters, while Figure 8 shows the clusters with respect to PCA.

Algorithm 4 Agglomerative hierarchical clustering

```
1. Initialize each sample $x_i \in X$ as a cluster singleton $C_i = x_i$ and compute the
   pairwise dissimilarity matrix $\Delta(C_i, C_j) = d(x_i, x_j), i \neq j$.
2. Repeat for $i = n, n-1, ..., 2$:
   (a) Among the $i$ clusters, merge the two clusters that have the lowest between-
       cluster dissimilarity $C_k = C_i \cup C_j$
   (b) Compute the new pairwise between-cluster dissimilarity among the $i-1$
       remaining clusters
           $$\Delta(C_k, C_l) = \underset{x_i \in C_k, x_j \in C_l}{max} d(x_i, x_j),\ k \leq l$$
```

```r
set.seed(123)
k_hc = hclust(dist(X), method = "complete")
hc_cluster = cutree(k_hc, k = 3)
```

Figure 7: Hierarchical clustering



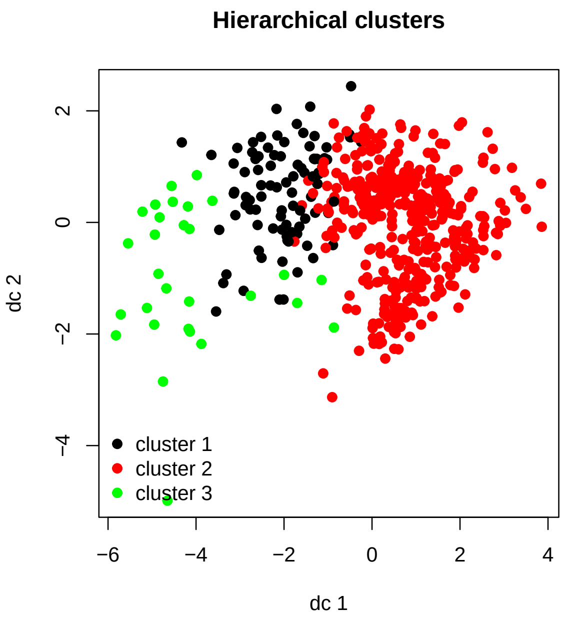
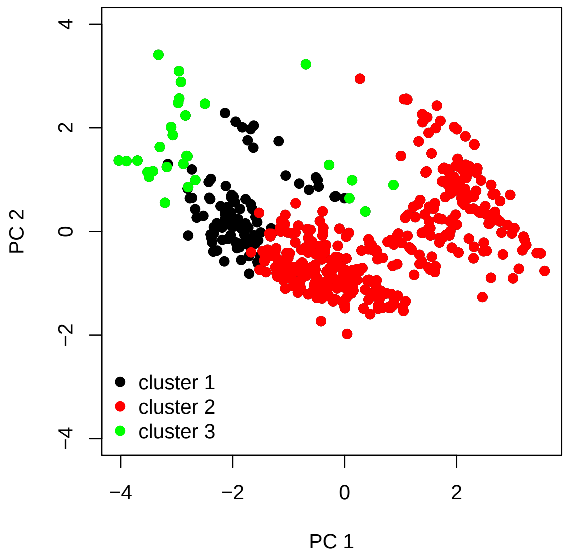


Figure 8: Hierarchical clustering vs PCA



Relative to the other methods, hierarchical clustering appears to have a higher number of classification errors with respect to expensive houeses. The algorithm allocates about 50% of expensive houses to cluster 3 and 43% to cluster 1. The clusters are very asimmetrical, with 76% of samples belonging to cluster 2 (Table 6).

Table 6: Hierarchical clustering

	cluster 1	cluster 2	cluster 3
number of houses	91	386	29
expensive houses	22	4	25
in %	43%	8%	49%

7 Cluster evalutation

The results of a clustering algorithm can be evaluated based on the following criteria:

- External criteria: a pre-specified structure, such as a set of data labels
- Internal criteria: the characteristics of the clustered data themselves or a comparison between different clusters
- Relative criteria: different clustering algorithms

The data is well classified if the clusters have the following properties:

- Compactness: the data belonging to a cluster are as close as possible (small within-cluster dissimilarity)
- Separation: the data belonging to different clusters are as far apart as possible (large between-cluster dissimilarity)

Due to the absence of unambiguous labels, in this exercise we will focus mainly on internal evaluation criteria.

7.1 Dunn index

The distance between clusters C_k and C_l, $k \neq l$ is measured by the distance between their closest points

$$d(C_k, C_l) = \min_{x_i \in C_k, x_j \in C_l} ||x_i - x_j||$$

The minimal distance between samples of different clusters is the smallest distance $d(C_k, C_l)$

$$d_{min} = \min_{k \neq l} \; d(C_k, C_l)$$

The diameter of the cluster is defined as the largest distance between two samples of the cluster. For a generic cluster C_k, the diameter is given by

$$D_k = \max_{x_i, x_j \in C_k, i \neq j} ||x_i - x_j||$$

The maximal within-cluster distance is the largest distance D_k

$$d_{max} = \max_{1 \leq k \leq K} D_k$$

The Dunn index is given by the ratio of the smallest between-cluster distance (separation) and the largest within-cluster distance (diameter)

$$D_K = \frac{d_{min}}{d_{max}}$$

The higher the Dunn index, the better the samples are classified. Table 7 shows a comparison of the Dunn index and a Dunn-like index computed as the ratio between the minimum average between-cluster dissimilarity and maximum average within-cluster dissimilarity. The indices are computed with the function `cluster.stats` from the package `fpc`. For the Dunn index, hierarchical clustering has the highest score, while for the Dunn-like index, GMM clustering has the highest score.

Table 7: Dunn index

	K-means	K-medoids	GMM	Hierarchical
Dunn index	0.02	0.03	0.05	0.06
Dunn-like index	1.36	1.38	1.51	1.12

7.2 Silhouette score

The silhouette score measures the classification error of the clustering algorithm using any distance metric. Its value ranges from -1 to 1 and measures how similar a sample is to its own cluster compared to other clusters.

The mean distance between a sample $x_i \in C_k$ and all other samples in the cluster is given by

$$a(x_i) = \frac{1}{|C_k| - 1} \sum_{x_j \in C_k, i \neq j} d(x_i, x_j)$$

where $|C_k|$ is the number of samples belonging to cluster k. A sample x_i is well classified if $a(x_i)$ is small and if the mean distance between $x_i \in C_k$ and any other cluster $C_l, k \neq l$ is small

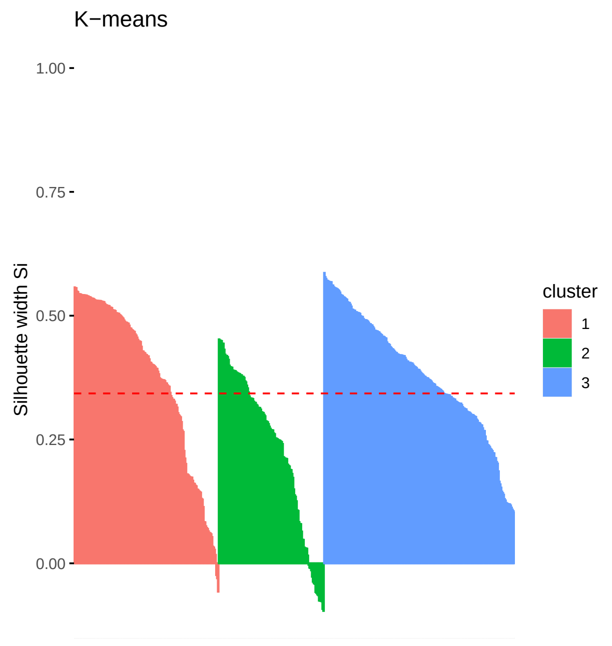
$$b(x_i) = \min_{k \neq l} \frac{1}{|C_k|} \sum_{x_j \in C_k} d(x_i, x_j)$$

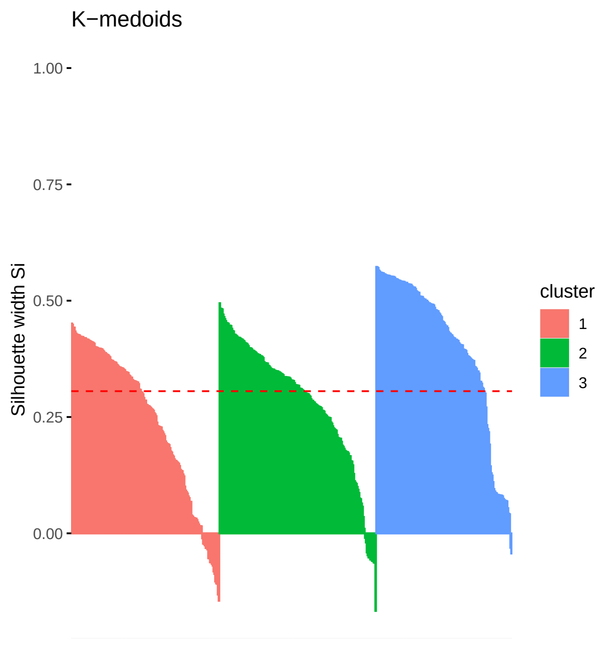
Hence, the silhouette value is defined as

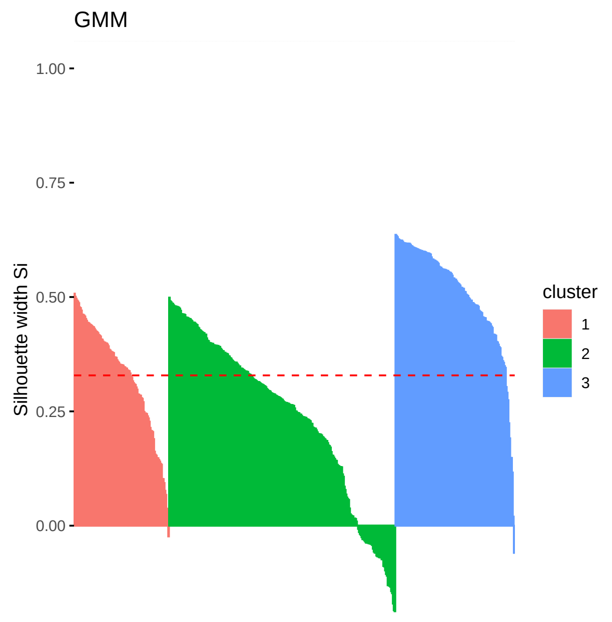
$$s(x_i) = \frac{b(x_i) - a(x_i)}{max\{a(x_i), b(x_i)\}}, \text{ if } |C_k| > 1$$
$$s(x_i) = 0, \text{ if } |C_k| = 1$$

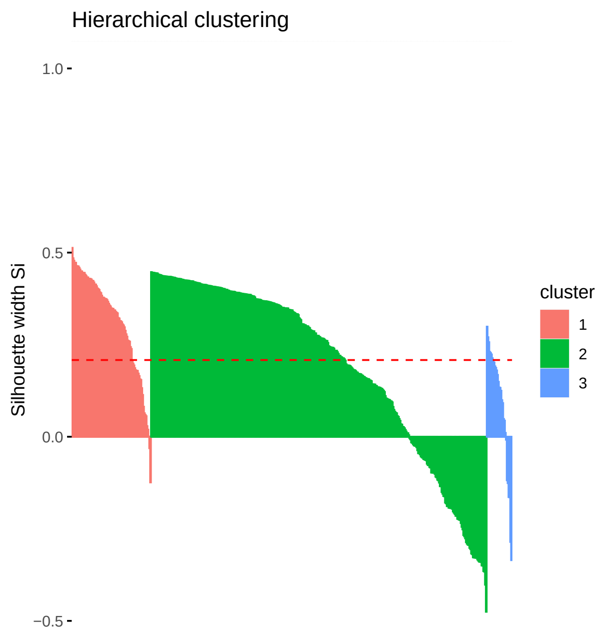
Figure 9 shows the silhouette score computed using the function `silhouette` from the package `cluster`. In this example, we use the Euclidean distance as a dissimilarity measure. Most of the samples are well classified, since their silhouette scores are above average (the red line). For all considered methods and especially for hierarchical clustering, most mis-classified samples belong to cluster 2 (negative silhouette score).

Figure 9: Silhouette score









8 References

Dunn, J. C. (1974). *Well-separated clusters and optimal fuzzy partitions.* Journal of cybernetics, 4(1), 95-104.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning (Vol. 1, No. 10).* New York: Springer series in statistics.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning (Vol. 112, p. 18).* New York: springer.

Harrison Jr, D., & Rubinfeld, D. L. (1978). *Hedonic housing prices and the demand for clean air.* Journal of environmental economics and management, 5(1), 81-102.

Kaufman, L., Rousseeuw, P.J. (1987). *Clustering by means of medoids.* In: Statistical Data Analysis Based on the L1 Norm and Related Methods, Y. Dodge (ed.), North-Holland, 405-416.

Kaufman, L., Rousseeuw, P.J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley & Sons.

Rentzmann, S., & Wuthrich, M. V. (2019). *Unsupervised Learning: What is a Sports Car?.* Available at SSRN 3439358.

Rousseeuw, P. J. (1987). *Silhouettes: a graphical aid to the interpretation and validation of cluster analysis.* Journal of computational and applied mathematics, 20, 53-65.

Schubert, E., & Rousseeuw, P. J. (2019, October). *Faster k-medoids clustering: improving the PAM, CLARA, and CLARANS algorithms.* In International conference on similarity search and applications (pp. 171-187). Springer, Cham.