# Chapter 6

# Clustering

This chapter presents examples of various clustering techniques in R, including $k$-means clustering, $k$-medoids clustering, hierarchical clustering and density-based clustering. The first two sections demonstrate how to use the $k$-means and $k$-medoids algorithms to cluster the *iris* data. The third section shows an example on hierarchical clustering on the same data. The last section describes the idea of density-based clustering and the DBSCAN algorithm, and shows how to cluster with DBSCAN and then label new data with the clustering model. For readers who are not familiar with clustering, introductions of various clustering techniques can be found in [Zhao et al., 2009a] and [Jain et al., 1999].

## 6.1 The k-Means Clustering

This section shows k-means clustering of `iris` data (see Section 1.3.1 for details of the data). At first, we remove species from the data to cluster. After that, we apply function `kmeans()` to `iris2`, and store the clustering result in `kmeans.result`. The cluster number is set to 3 in the code below.

```
> iris2 <- iris
> iris2$Species <- NULL
> (kmeans.result <- kmeans(iris2, 3))

K-means clustering with 3 clusters of sizes 38, 50, 62

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.006000    3.428000     1.462000    0.246000
3     5.901613    2.748387     4.393548    1.433871

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [40] 2 2 2 2 2 2 2 2 2 2 2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1
 [79] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1 1 3 3 1 1
[118] 1 1 3 1 3 1 3 1 1 3 3 1 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1 1 1 3 1 1 1 3 1 1 3

Within cluster sum of squares by cluster:
[1] 23.87947 15.15100 39.82097
 (between_SS / total_SS =  88.4 %)

Available components:
```

```
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"
```

The clustering result is then compared with the class label (`Species`) to check whether similar objects are grouped together.

```
> table(iris$Species, kmeans.result$cluster)

            1  2  3
  setosa    0 50  0
  versicolor  2  0 48
  virginica  36  0 14
```

The above result shows that cluster "setosa" can be easily separated from the other clusters, and that clusters "versicolor" and "virginica" are to a small degree overlapped with each other.

Next, the clusters and their centers are plotted (see Figure 6.1). Note that there are four dimensions in the data and that only the first two dimensions are used to draw the plot below. Some black points close to the green center (asterisk) are actually closer to the black center in the four dimensional space. We also need to be aware that the results of k-means clustering may vary from run to run, due to random selection of initial cluster centers.

```
> plot(iris2[c("Sepal.Length", "Sepal.Width")], col = kmeans.result$cluster)
> # plot cluster centers
> points(kmeans.result$centers[,c("Sepal.Length", "Sepal.Width")], col = 1:3,
+        pch = 8, cex=2)
```

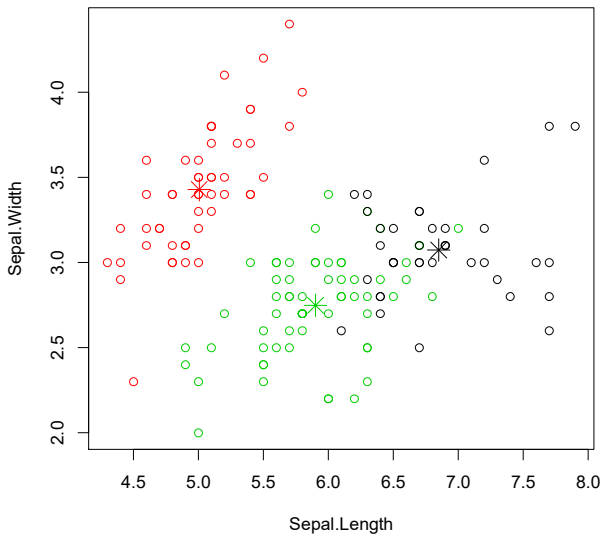


Figure 6.1: Results of k-Means Clustering

More examples of $k$-means clustering can be found in Section 7.3 and Section 10.8.1.

## 6.2 The k-Medoids Clustering

This sections shows k-medoids clustering with functions `pam()` and `pamk()`. The k-medoids clustering is very similar to k-means, and the major difference between them is that: while a cluster is represented with its center in the k-means algorithm, it is represented with the object closest to the center of the cluster in the k-medoids clustering. The k-medoids clustering is more robust than k-means in presence of outliers. PAM (Partitioning Around Medoids) is a classic algorithm for k-medoids clustering. While the PAM algorithm is inefficient for clustering large data, the CLARA algorithm is an enhanced technique of PAM by drawing multiple samples of data, applying PAM on each sample and then returning the best clustering. It performs better than PAM on larger data. Functions `pam()` and `clara()` in package *cluster* [Maechler et al., 2012] are respectively implementations of PAM and CLARA in R. For both algorithms, a user has to specify $k$, the number of clusters to find. As an enhanced version of `pam()`, function `pamk()` in package *fpc* [Hennig, 2010] does not require a user to choose $k$. Instead, it calls the function `pam()` or `clara()` to perform a partitioning around medoids clustering with the number of clusters estimated by optimum average silhouette width.

With the code below, we demonstrate how to find clusters with `pam()` and `pamk()`.

```
> library(fpc)
> pamk.result <- pamk(iris2)
> # number of clusters
> pamk.result$nc
```

```
[1] 2
```

```
> # check clustering against actual species
> table(pamk.result$pamobject$clustering, iris$Species)
```

```
   setosa versicolor virginica
1      50          1         0
2       0         49        50
```

```
> layout(matrix(c(1,2),1,2)) # 2 graphs per page
> plot(pamk.result$pamobject)
> layout(matrix(1)) # change back to one graph per page
```

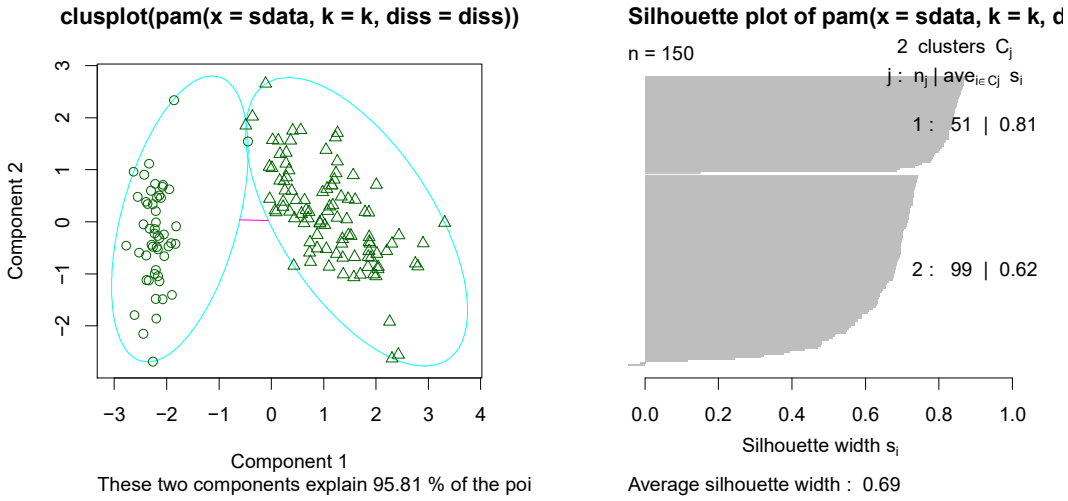


Figure 6.2: Clustering with the $k$-medoids Algorithm - I

In the above example, `pamk()` produces two clusters: one is "setosa", and the other is a mixture of "versicolor" and "virginica". In Figure 6.2, the left chart is a 2-dimensional "clusplot" (clustering plot) of the two clusters and the lines show the distance between clusters. The right one shows their silhouettes. In the silhouette, a large $s_i$ (almost 1) suggests that the corresponding observations are very well clustered, a small $s_i$ (around 0) means that the observation lies between two clusters, and observations with a negative $s_i$ are probably placed in the wrong cluster. Since the average $S_i$ are respectively 0.81 and 0.62 in the above silhouette, the identified two clusters are well clustered.

Next, we try `pam()` with $k = 3$.

```
> pam.result <- pam(iris2, 3)
> table(pam.result$clustering, iris$Species)
```

```
   setosa versicolor virginica
1      50          0         0
2       0         48        14
3       0          2        36
```

```
> layout(matrix(c(1,2),1,2)) # 2 graphs per page
> plot(pam.result)
> layout(matrix(1)) # change back to one graph per page
```

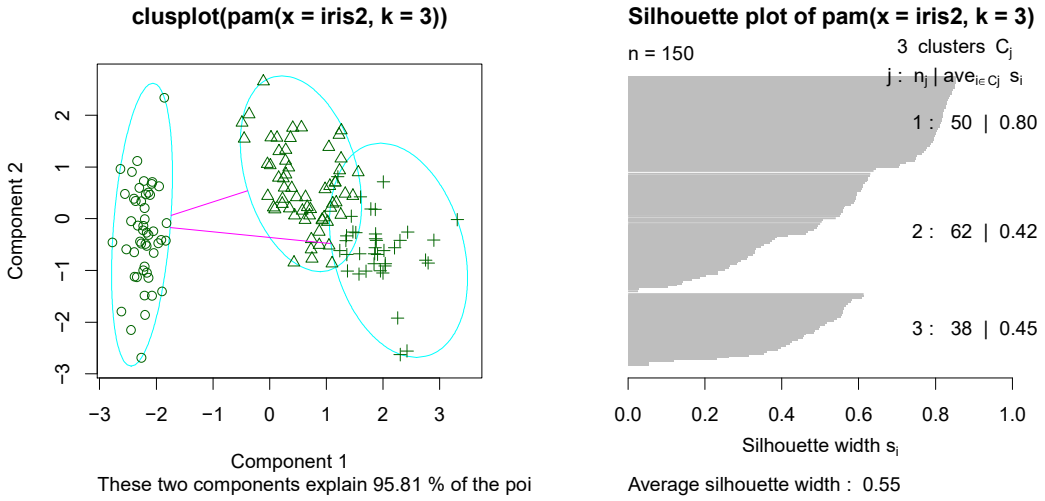


Figure 6.3: Clustering with the $k$-medoids Algorithm - II

With the above result produced with `pam()`, there are three clusters: 1) cluster 1 is species "setosa" and is well separated from the other two; 2) cluster 2 is mainly composed of "versicolor", plus some cases from "virginica"; and 3) the majority of cluster 3 are "virginica", with two cases from "versicolor".

It's hard to say which one is better out of the above two clusterings produced respectively with `pamk()` and `pam()`. It depends on the target problem and domain knowledge and experience. In this example, the result of `pam()` seems better, because it identifies three clusters, corresponding to three species. Therefore, the heuristic way to identify the number of clusters in `pamk()` does not necessarily produce the best result. Note that we cheated by setting $k = 3$ when using `pam()`, which is already known to us as the number of species.

More examples of $k$-medoids clustering can be found in Section 10.8.2.

## 6.3 Hierarchical Clustering

This section demonstrates hierarchical clustering with `hclust()` on `iris` data (see Section 1.3.1 for details of the data).

We first draw a sample of 40 records from the iris data, so that the clustering plot will not be over crowded. Same as before, variable `Species` is removed from the data. After that, we apply hierarchical clustering to the data.

```
> idx <- sample(1:dim(iris)[1], 40)
> irisSample <- iris[idx,]
> irisSample$Species <- NULL
> hc <- hclust(dist(irisSample), method="ave")
```

```
> plot(hc, hang = -1, labels=iris$Species[idx])
> # cut tree into 3 clusters
> rect.hclust(hc, k=3)
> groups <- cutree(hc, k=3)
```

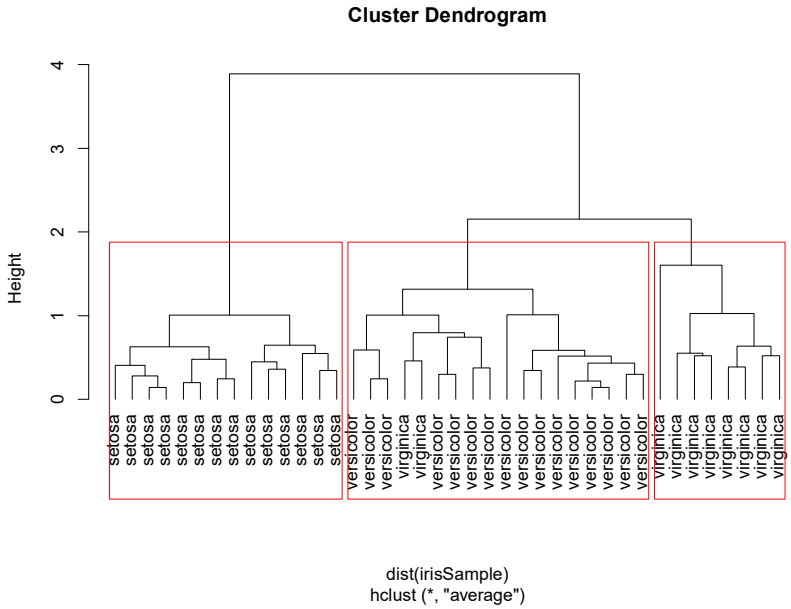


Figure 6.4: Cluster Dendrogram

Similar to the above clustering of $k$-means, Figure 6.4 also shows that cluster "setosa" can be easily separated from the other two clusters, and that clusters "versicolor" and "virginica" are to a small degree overlapped with each other.

More examples of hierarchical clustering can be found in Section 8.4 and Section 10.7.

## 6.4 Density-based Clustering

The DBSCAN algorithm [Ester et al., 1996] from package *fpc* [Hennig, 2010] provides a density-based clustering for numeric data. The idea of density-based clustering is to group objects into one cluster if they are connected to one another by densely populated area. There are two key parameters in DBSCAN :

- `eps`: reachability distance, which defines the size of neighborhood; and
- `MinPts`: minimum number of points.

If the number of points in the neighborhood of point $\alpha$ is no less than `MinPts`, then $\alpha$ is a *dense point*. All the points in its neighborhood are *density-reachable* from $\alpha$ and are put into the same cluster as $\alpha$.

The strengths of density-based clustering are that it can discover clusters with various shapes and sizes and is insensitive to noise. As a comparison, the $k$-means algorithm tends to find clusters with sphere shape and with similar sizes.

Below is an example of density-based clustering of the `iris` data.

```
> library(fpc)
> iris2 <- iris[-5] # remove class tags
```

```
> ds <- dbscan(iris2, eps=0.42, MinPts=5)
> # compare clusters with original class labels
> table(ds$cluster, iris$Species)
```

```
   setosa versicolor virginica
0       2         10        17
1      48          0         0
2       0         37         0
3       0          3        33
```

In the above table, "1" to "3" in the first column are three identified clusters, while "0" stands for noises or outliers, i.e., objects that are not assigned to any clusters. The noises are shown as black circles in Figure 6.5.

```
> plot(ds, iris2)
```

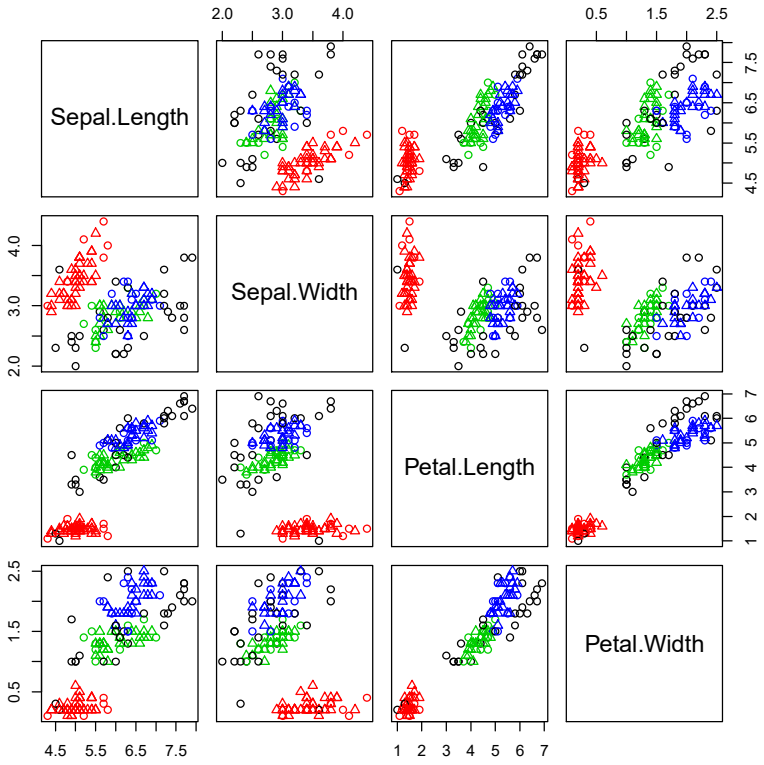


Figure 6.5: Density-based Clustering - I

The clusters are shown below in a scatter plot using the first and fourth columns of the data.

```
> plot(ds, iris2[c(1,4)])
```

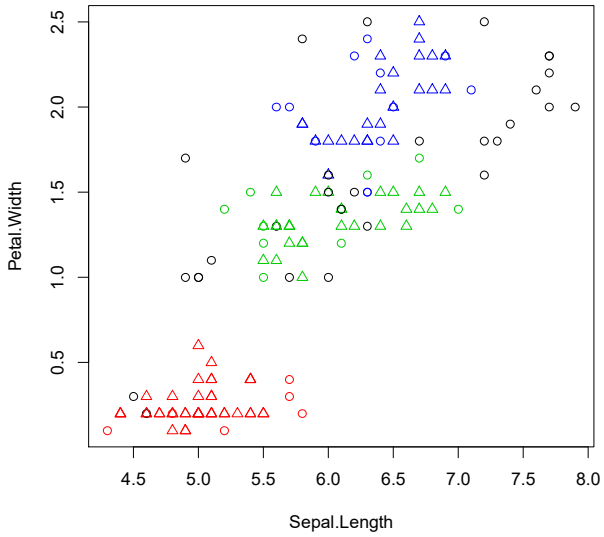


Figure 6.6: Density-based Clustering - II

Another way to show the clusters is using function `plotcluster()` in package *fpc*. Note that the data are projected to distinguish classes.

```
> plotcluster(iris2, ds$cluster)
```

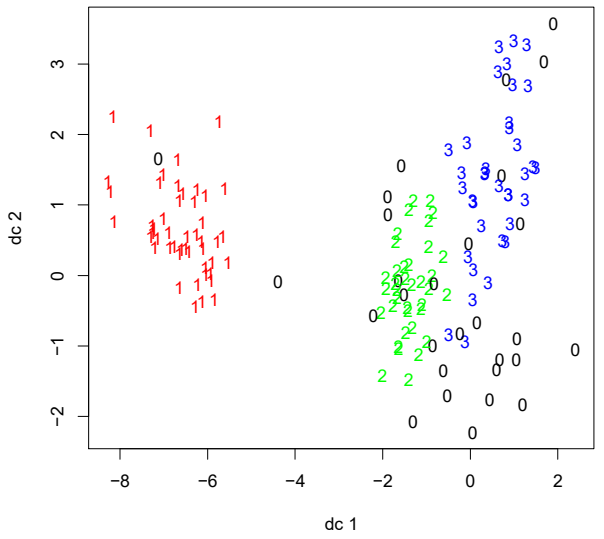


Figure 6.7: Density-based Clustering - III

The clustering model can be used to label new data, based on the similarity between new data and the clusters. The following example draws a sample of 10 objects from `iris` and adds small noises to them to make a new dataset for labeling. The random noises are generated with a uniform distribution using function `runif()`.

```
> # create a new dataset for labeling
> set.seed(435)
> idx <- sample(1:nrow(iris), 10)
> newData <- iris[idx,-5]
> newData <- newData + matrix(runif(10*4, min=0, max=0.2), nrow=10, ncol=4)
> # label new data
> myPred <- predict(ds, iris2, newData)
> # plot result
> plot(iris2[c(1,4)], col=1+ds$cluster)
> points(newData[c(1,4)], pch="*", col=1+myPred, cex=3)
> # check cluster labels
> table(myPred, iris$Species[idx])

myPred setosa versicolor virginica
     0      0          0         1
     1      3          0         0
     2      0          3         0
     3      0          1         2
```

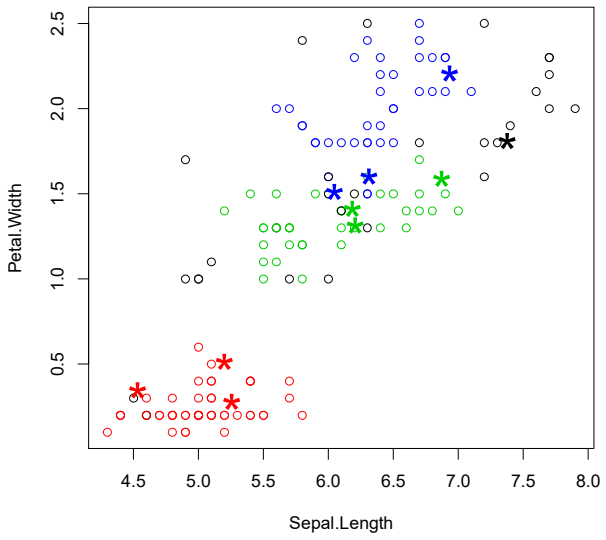


Figure 6.8: Prediction with Clustering Model

As we can see from the above result, out of the 10 new unlabeled data, 8(=3+3+2) are assigned with correct class labels. The new data are shown as asterisk("*") in the above figure and the colors stand for cluster labels in Figure 6.8.