Clustering - IV - LAB

Prof. Dan A. Simovici

UMB
1 Partitional Clustering in R
The function `kmeans` of R performs the $k$-means clustering on a data matrix. The standard usage of this function is

```r
kmeans(D, centers, iter.max = 10, nstart = 1, algorithm,trace)
```
The Arguments of kmeans:

- D: a matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns);
- centers: either the number of clusters, say \( k \), or a set of initial (distinct) cluster centroids. In the first case, a random set of (distinct) rows in \( D \) is chosen as the set of initial centroids;
- iter.max: the maximum number of iterations allowed;
- nstart: if centers is a number, this parameter indicates the number of random sets;
- algorithm: a string that indicates the variant of the algorithm, as discussed previously;
- trace: logical or integer number, currently only used in the default method ("Hartigan-Wong"): if positive (or true), tracing information on the progress of the algorithm is produced; higher values may produce more tracing information.
The function `kmeans` returns an object which has a print and a fitted method. It is a list with at least the following components:

- `cluster`: a vector of integers (from 1 to $k$) indicating the cluster to which each point is allocated;
- `centers`: a matrix of cluster centres;
- `totss`: the total sum of squares;
- `withinss`: vector of within-cluster sum of squares, one component per cluster;
- `tot.withinss`: total within-cluster sum of squares, that is, the sum(withinss);
- `betweenss`: the between-cluster sum of squares;
- `size`: the number of points in each cluster;
- `iter`: the number of (outer) iterations;
Example

To generate a set of $n$ points in $\mathbb{R}^2$ normally distributed around the vector `center` and having the standard deviations specified by the vector `stdev` we use the function `setofpoints2` that produces a set of 2-dimensional set of points. This function is defined as:

```r
setofpoints2 <- function(n, center, stdev) {
  return(cbind(rnorm(n, center[1], stdev[1]),
               rnorm(n, center[2], stdev[2])))
}
```
The sets of points A, B and C are generated and, then joined into the set D; the columns of the matrix D are named as ("x") and ("y"):

```r
A <- setofpoints2(40,c(2,4),c(0.2,0.3))
B <- setofpoints2(30,c(3,2.5),c(0.3,0.3))
C <- setofpoints2(45,c(4,4),c(0.2,0.4))
D <- rbind(A,B,C)
colnames(D) <- c("x","y")
```
The plot of $D$ is shown below:
Example

kmeans applied to the data set D produces the object sp:

```r
sp <- kmeans(D,3)
plot(D,col=sp\$cluster)
```
The function `pam`, a component of the package `cluster` implements the algorithm discussed above. To apply the algorithm to the data set `D` previously computed we write:

```
pamx <- pam(D, 3)
```
The summary of the object pamx returned by summary(pamx) is

Medoids: ID x y
Clustering vector: [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[38] 1 1 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
[75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

Objective function: build swap 0.4756183 0.3719818

Available components: [1] "medoids" "id.med" "clustering" "objective" "isolation" [6] "clusinfo" "silinfo" "diss" "call" "data"
Clusters can be visualized using the function `clusplot` which result in the representation shown next.

```
clusplot(pam(x = D, k = 3))
```

These two components explain 100% of the point variability.
Example

This data set describes 50 specimens from each of the three subspecies: *Iris Setosa*, *Iris Versicolor* and *Iris Virginica*. The data gives the measurements in centimeters of the variables *Sepal.Length*, *Sepal.width*, *Petal.length* and *Petal.Width* for each of the specimens. The structure of the *iris* data set is described below:

```r
> str(iris)
'data.frame': 150 obs. of 5 variables:
$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ... 
$ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ... 
$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ... 
$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ... 
$ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 ...
```
To cluster this data set we need to eliminate the Species factor, which can done by

```r
iris[, -5]
```

The variance of the remaining four components can be computed by writing

```r
> sapply(iris[, -5], var)
```

and returns the result

<table>
<thead>
<tr>
<th>Sepal.Length</th>
<th>Sepal.Width</th>
<th>Petal.Length</th>
<th>Petal.Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6856935</td>
<td>0.1899794</td>
<td>3.1162779</td>
<td>0.5810063</td>
</tr>
</tbody>
</table>
The package ggplot2 is used to present the results of the clustering that takes into account the attributes Petal.Length and Petal.Width and is invoked using

```r
> library(ggplot2)
```

The results are displayed using

```r
> ggplot(iris,aes(x=Petal.Length,y=Petal.Width,col=Species)) + geom_point()
```
The results are shown next.
Since the $k$-means algorithm is parametrized by $k$, finding the optimum number of clusters requires a heuristic approach. One of the common method to determine $k$ is to examine the percentage of variance explained as a function of the number of clusters. The idea is to choose $k$ so that adding another cluster does not produce a much better modeling of the data. If one plots the total variance against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the *elbow criterion*. Identifying this *elbow* cannot always be done unambiguously.
The plot of within cluster sum of squares versus the number of clusters indicates an elbow point at 3, which suggests that 3 is the optimum for $k$. 
The code that allows the construction of this plot is given next.

```r
set.seed(200)
k.max <- 10
wss <- sapply(1:k.max, 
               function(k){kmeans(iris[,3:4],k,nstart = 20,iter.max = 20)$tot.withinss})
wss
plot(1:k.max,wss, type= "b",
     xlab = "Number of clusters(k)",
     ylab = "Within cluster sum of squares")
```

The final model is built using `kmeans` and $k = 3$. The `nstart` value has is defined as 20 which means that R will try 20 different random starting assignments and then select the one with the lowest within cluster variation.

```r
icluster <- kmeans(iris[,3:4],3,nstart = 20)
table(icluster$cluster,iris$Species)
```

```
##
## setosa  versicolor  virginica
## 1       0          2      46
## 2       50         0       0
## 3       0          48     4
```

Note that for reproductibility we defined a seed value.