Chapter 13
k-Means Clustering

As we learned in Chaps. 7, 8, and 9, classification could help us make predictions on new observations. However, classification requires (human supervised) predefined label classes. What if we are in the early phases of a study and/or don't have the required resources to manually define, derive or generate these class labels?

Clustering can help us explore the dataset and separate cases into groups representing similar traits or characteristics. Each group could be a potential candidate for a class. Clustering is used for exploratory data analytics, i.e., as *unsupervised learning*, rather than for confirmatory analytics or for predicting specific outcomes.

In this chapter, we will present (1) clustering as a machine learning task, (2) the *silhouette* plots for classification evaluation, (3) the *k-Means* clustering algorithm and how to tune it, (4) examples of several interesting case-studies, including Divorce and Consequences on Young Adults, Pediatric Trauma, and Youth Development, (5) demonstrate hierarchical clustering, and (6) Gaussian mixture modeling.

13.1 Clustering as a Machine Learning Task

As we mentioned before, clustering represents classification of unlabeled cases. Scatter plots depict a simple illustration of the clustering process. Assume we don't know much about the ingredients of frankfurter hot dogs and we have the following graph (Fig. 13.1).

I. D. Dinov, *Data Science and Predictive Analytics*,
https://doi.org/10.1007/978-3-319-72347-1_13

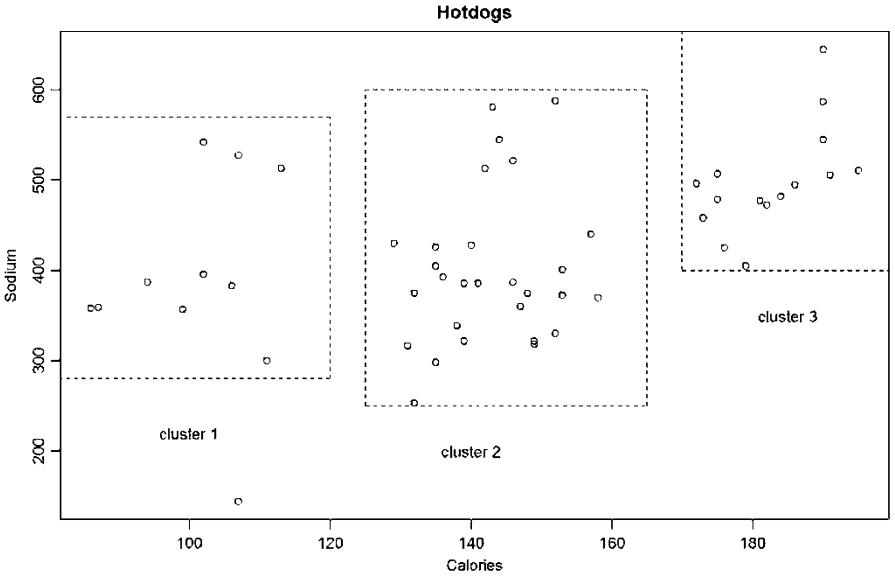

Fig. 13.1 Hotdogs dataset – scatterplot of calories and sodium content blocked by type of meat

```
# See Chapter 12 code for complete details
# install.packages("rvest")
library(rvest)
wiki_url<-
read_html("http://wiki.socr.umich.edu/index.php/SOCR_012708_ID_Data_HotDogs"
)
html_nodes(wiki_url, "#content")
hotdog<- html_table(html_nodes(wiki_url, "table")[[1]])
plot(hotdog$Calories, hotdog$Sodium, main = "Hotdogs", xlab="Calories",
ylab="Sodium")
```

In terms of calories and sodium, these hot dogs are clearly separated into three different clusters. *Cluster 1* has hot dogs of low calories and medium sodium content; *Cluster 2* has both calorie and sodium at medium levels; *Cluster 3* has both sodium and calories at high levels. We can make a bold guess about the meats used in these three clusters of hot dogs. For Cluster 1, it could be mostly chicken meat since it has low calories. The second cluster might be beef, and the third one is likely to be pork, because beef hot dogs have considerably less calories and salt than pork hot dogs. However, this is just guessing. Some hot dogs have a mixture of two or three types of meat. The real situation is somewhat similar to what we guessed but with some random noise, especially in Cluster 2.

The following two plots show the primary type of meat used for each hot dog labeled by name (top) and color-coded (bottom) (Figs. 13.2 and 13.3).

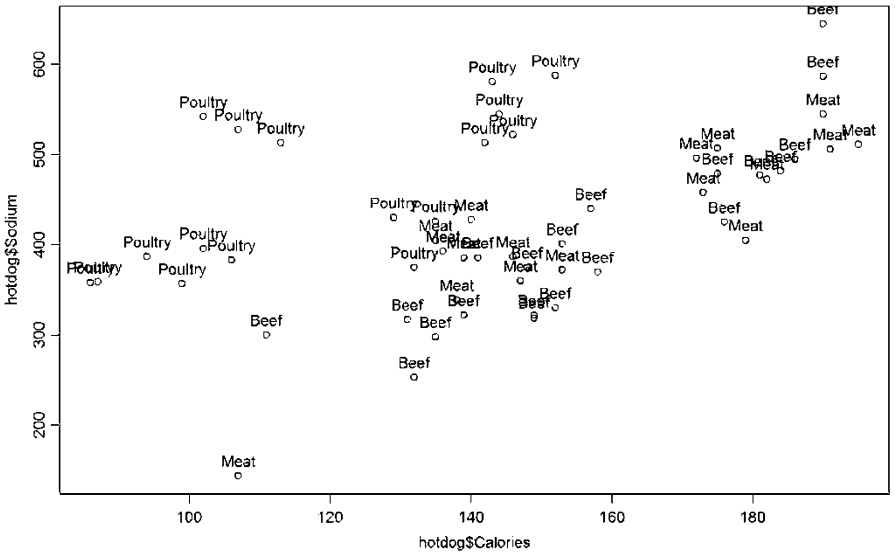

Fig. 13.2 Scatterplot of calories and sodium content with meat type labels

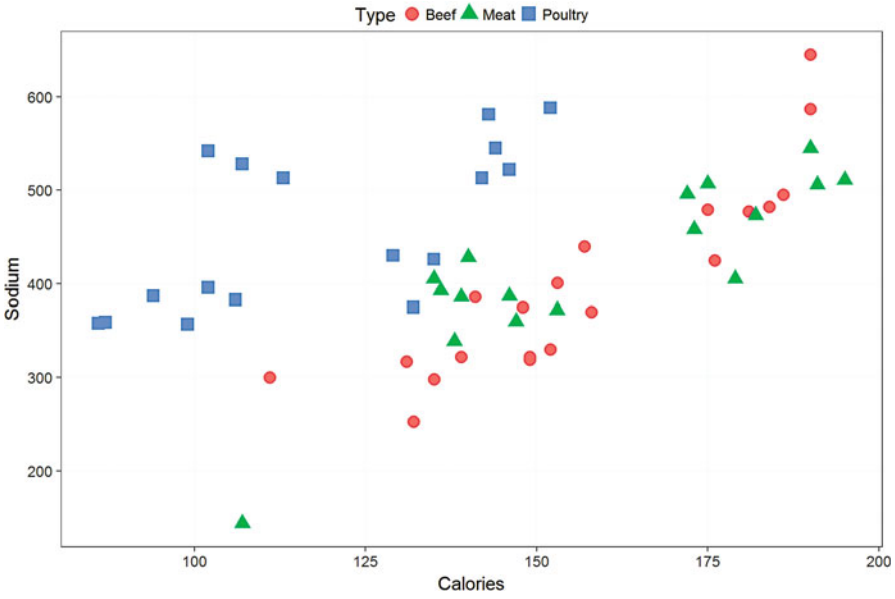

Fig. 13.3 An alternative scatterplot of the hotdogs calories and sodium

13.2 Silhouette Plots

Silhouette plots are useful for interpretation and validation of consistency of all clustering algorithms. The silhouette value, $\in[-1, 1]$, measures the similarity (cohesion) of a data point to its cluster relative to other clusters (separation). Silhouette plots rely on a distance metric, e.g., the Euclidean distance, Manhattan distance, Minkowski distance, etc.

- High silhouette value suggest that the data matches its own cluster well.
- A clustering algorithm performs well when most Silhouette values are high.
- Low value indicates poor matching within the neighboring cluster.
- Poor clustering may imply that the algorithm configuration may have too many or too few clusters.

Suppose a clustering method groups all data points (objects), $\{X_i\}_i$, into k clusters and define:

- d_i as the *average dissimilarity* of X_i with all other data points within its cluster. d_i captures the quality of the assignment of X_i to its current class label. Smaller or larger d_i values suggest better or worse overall assignment for X_i to its cluster, respectively. The average dissimilarity of X_i to a cluster C is the average distance between X_i and all points in the cluster of points labeled C.
- l_i as the *lowest average dissimilarity* of X_i to any other cluster, that X_i is not a member of. The cluster corresponding to l_i, the lowest average dissimilarity, is called the X_i **neighboring cluster**, as it is the next best fit cluster for X_i.

Then, the **silhouette** of X_i is defined by:

$$-1 \leq s_i = \frac{l_i - d_i}{\max\{l_i, d_i\}} \equiv \begin{cases} 1 - \dfrac{d_i}{l_i}, & \text{if } d_i < l_i \\ 0, & \text{if } d_i = l_i \\ \dfrac{l_i}{d_i} - 1, & \text{if } d_i > l_i \end{cases}.$$

Note that:

- $-1 \leq s_i \leq 1$,
- $s_i \to 1$ when $d_i \ll l_i$, i.e., the dissimilarity of X_i to its cluster, C is much lower relative to its dissimilarity to other clusters, indicating a good (cluster assignment) match. Thus, high Silhouette values imply the data is appropriately clustered.
- Conversely, $-1 \leftarrow s_i$ when $l_i \ll d_i$, d_i is large, implying a poor match of X_i with its current cluster C, relative to neighboring clusters. X_i may be more appropriately clustered in its neighboring cluster.
- $s_i \sim 0$ means that the X_i may lie on the border between two natural clusters.

13.3 The k-Means Clustering Algorithm

The *k-means* algorithm is one of the most commonly used algorithms for clustering.

13.3.1 Using Distance to Assign and Update Clusters

This algorithm is similar to *k-nearest neighbors (KNN)* presented in Chap. 7. In clustering, we don't have a priori pre-determined labels, and the algorithm is trying to deduce intrinsic groupings in the data.

Similar to KNN, k-means uses Euclidean distance ($\|_2$ norm) most of the times, however Manhattan distance ($\|_1$ norm), or the more general Minkowski distance $\left(\left(\sum_{i=1}^{n} |p_i - q_i|^c\right)^{\frac{1}{c}}\right)$ may also be used. For $c = 2$, the Minkowski distance represents the classical Euclidean distance:

$$dist(x, y) = \sqrt{\sum_{n=1}^{n} (x_i - y_i)^2}.$$

How can we separate clusters using this formula? The *k-means* protocol is as follows:

- *Initiation*: First, we define k points as cluster centers. Often these points are k random points from the dataset. For example, if $k = 3$, we choose three random points in the dataset as cluster centers.
- *Assignment*: Second, we determine the maximum extent of the cluster boundaries that all have maximal distance from their cluster centers. Now the data is separated into k initial clusters. The assignment of each observation to a cluster is based on computing the least within-cluster sum of squares according to the chosen distance. Mathematically, this is equivalent to Voronoi tessellation of the space of the observations according to their mean distances.
- *Update*: Third, we update the centers of our clusters to new *means* of the cluster centroid locations. This updating phase is the essence of the *k-means* algorithm.

Although there is no guarantee that the *k-means* algorithm converges to a global optimum, in practice, the algorithm tends to converge, i.e., the assignments no longer change, to a local minimum as there are only a finite number of such Voronoi partitionings.

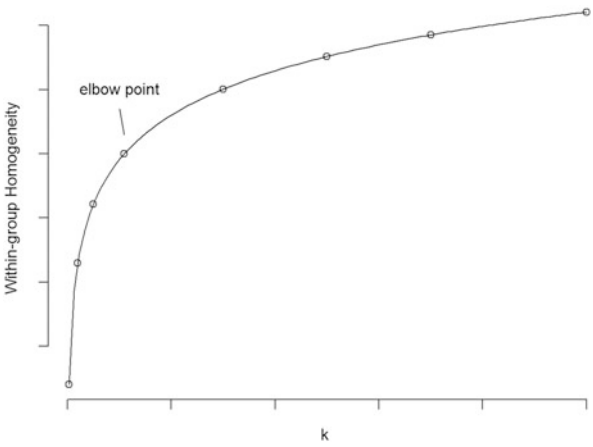

Fig. 13.4 Elbow plot of the within-group homogeneity against the number of groups parameter (k)

13.3.2 Choosing the Appropriate Number of Clusters

We don't want our number of clusters to be either too large or too small. If it is too large, the groups are too specific to be meaningful. On the other hand, too few groups might be too broadly general to be useful. As we mentioned in Chap. 7, $k = \sqrt{\frac{n}{2}}$ is a good place to start. However, it might generate a large number of groups. Also, the elbow method may be used to determine the relationship of k and homogeneity of the observations of each cluster. When we graph within-group homogeneity against k, we can find an "elbow point" that suggests a minimum k corresponding to relatively large within-group homogeneity (Fig. 13.4).

This graph shows that homogeneity barely increases above the "elbow point". There are various ways to measure homogeneity within a cluster. For detailed explanations please read On clustering validation techniques, *Journal of Intelligent Information Systems* Vol. 17, pp. 107–145, by M. Halkidi, Y. Batistakis, and M. Vazirgiannis (2001).

13.4 Case Study 1: Divorce and Consequences on Young Adults

13.4.1 Step 1: Collecting Data

The dataset we will be using is the Divorce and Consequences on Young Adults dataset. This is a longitudinal study focused on examining the consequences of recent parental divorce for young adults (initially ages 18–23) whose parents had divorced within 15 months of the study's first wave (1990–91). The sample consisted of 257 White respondents with newly divorced parents. Here we have a subset of this dataset with 47 respondents in our case-studies folder, CaseStudy01_Divorce_YoungAdults_Data.csv.

Variables

- **DIVYEAR**: Year in which parents were divorced. Dichotomous variable with 1989 and 1990.
- **Child affective relations**:
  - Momint: Mother intimacy. Interval level data with four possible responses (1-extremely close, 2-quite close, 3-fairly close, 4- not close at all).
  - Dadint: Father intimacy. Interval level data with four possible responses (1-extremely close, 2-quite close, 3-fairly close, 4-not close at all).
  - Live with mom: Polytomous variable with three categories (1- mother only, 2-father only, 3- both parents).
- **momclose**: measure of how close the child is to the mother (1-extremely close, 2-quite close, 3-fairly close, 4-not close at all).
- **Depression**: Interval level data regarding feelings of depression in the past 4 weeks. Possible responses are 1-often, 2-sometimes, 3-hardly ever, 4-never.
- **Gethitched**: Polytomous variable with four possible categories indicating respondent's plan for marriage (1-Marry fairly soon, 2-marry sometime, 3-never marry, 8-don't know).

13.4.2 Step 2: Exploring and Preparing the Data

Let's load the dataset and pull out a summary of all variables.

```
divorce<-read.csv("https://umich.instructure.com/files/399118/download?downl
oad_frd=1")
summary(divorce)
```

```
##     DIVYEAR          momint          dadint         momclose
##  Min.   :89.00   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:89.00   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.000
##  Median :90.00   Median :1.000   Median :2.000   Median :2.000
##  Mean   :89.68   Mean   :1.809   Mean   :2.489   Mean   :1.809
##  3rd Qu.:90.00   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:2.000
##  Max.   :90.00   Max.   :4.000   Max.   :4.000   Max.   :4.000
##    depression     livewithmom      gethitched
##  Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:2.000
##  Median :3.000   Median :1.000   Median :2.000
##  Mean   :2.851   Mean   :1.489   Mean   :2.213
##  3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :4.000   Max.   :9.000   Max.   :8.000
```

According to the summary, DIVYEAR is actually a dummy variable (either 89 or 90). We can recode (binarize) the DIVYEAR using the `ifelse()` function (mentioned in Chap. 8). The following line of code generates a new indicator variable for *divorce year = 1990*.

```
divorce$DIVYEAR<-ifelse(divorce$DIVYEAR==89, 0, 1)
```

We also need another preprocessing step to deal with `livewithmom`, which has missing values, `livewithmom = 9`. We can impute these using `momint` and `dadint` variables for each specific participant

```
table(divorce$livewithmom)

##
##  1  2  9
## 31 15  1

divorce[divorce$livewithmom==9, ]

##    DIVYEAR momint dadint momclose depression livewithmom gethitched
## 45       1      3      1        3          3           9          2
```

For instance, respondents that feel much closer to their dads may be assigned `divorce$livewithmom==2`, suggesting they most likely live with their fathers. Of course, alternative imputation strategies are also possible.

```
divorce[45, 6]<-2
divorce[45, ]

##    DIVYEAR momint dadint momclose depression livewithmom gethitched
## 45       1      3      1        3          3           2          2
```

13.4.3 Step 3: Training a Model on the Data

We are only using R base functionality, so no need to install any additional packages now, however `library(stats)` may still be necessary. Then, the function `kmeans()` will provide the *k-means* clustering of the data.

```
myclusters<-kmeans(mydata, k)
```

- *mydata*: dataset in a matrix form.
- *k*: number of clusters we want to create.
- *output*:
  - myclusters$cluster: vector indicating the cluster number for every observation.
  - myclusters$center: a matrix showing the mean feature values for every center.
  - mycluster$size: a table showing how many observations are assigned to each cluster.

Before we perform clustering, we need to standardize the features to avoid biasing the clustering based on features that use large-scale values. Note that distance calculations are sensitive to measuring units. The method `as.data.frame()` will convert our dataset into a data frame allowing us to use the `lapply()` function. Next, we use a combination of `lapply()` and `scale()` to standardize our data.

```
di_z<- as.data.frame(lapply(divorce, scale))
str(di_z)

## 'data.frame':    47 obs. of  7 variables:
##  $ DIVYEAR    : num  0.677 0.677 -1.445 0.677 -1.445 ...
##  $ momint     : num  1.258 1.258 -0.854 1.258 -0.854 ...
##  $ dadint     : num  -0.514 -0.514 0.536 1.586 0.536 ...
##  $ momclose   : num  0.225 1.401 -0.951 1.401 0.225 ...
##  $ depression : num  0.164 -0.937 1.265 0.164 -2.038 ...
##  $ livewithmom: num  -0.711 1.377 -0.711 -0.711 -0.711 ...
##  $ gethitched : num  0.846 -0.229 -0.229 0.846 -0.229 ...
```

The resulting dataset, `di_z`, is standardized so all features are unitless and follow approximately standardized normal distribution.

Next, we need to think about selecting a proper k. We have a relatively small dataset with 47 observations. Obviously we cannot have a k as large as 10. The rule of thumb suggests $k = \sqrt{47/2} = 4.8$. This would be relatively large also because we will have less than 10 observations for each cluster. It is very likely that for some clusters we only have one observation. A better choice may be 3. Let's see if this will work.

```
library(stats)
set.seed(321)
diz_clussters<-kmeans(di_z, 3)
```

13.4.4 Step 4: Evaluating Model Performance

Let's look at the clusters created by the *k-means* model.

```
diz_clussters$size
## [1] 12 24 11
```

At first glance, it seems that 3 worked well for the number of clusters. We don't have any cluster that contains a small number of observations. The three clusters have relatively equal number of respondents.

Silhouette plots represent the most appropriate evaluation strategy to assess the quality of the clustering. Silhouette values are between -1 and 1. In our case, two data points correspond to negative Silhouette values, suggesting these cases may be "mis-clustered" or perhaps are ambiguous, as the Silhouette value is close to 0. We can observe that the average Silhouette is reasonable, about 0.2 (Fig. 13.5).

Fig. 13.5 Silhouette plot for the 3 classes

```
require(cluster)

dis = dist(di_z)
sil = silhouette(diz_clussters$cluster, dis)
summary(sil)

## Silhouette of 47 units in 3 clusters from silhouette.default(x = diz_clus
sters$cluster, dist = dis) :
##  Cluster sizes and average silhouette widths:
##         12         24         11
## 0.16444649 0.27684356 0.07921684
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.08466  0.11760  0.20080  0.20190  0.30450  0.39820

plot(sil)
```

The next step would be to interpret the clusters in the context of this social study.

```
diz_clussters$centers

##      DIVYEAR     momint      dadint   momclose depression livewithmom
## 1  0.5004720  1.1698438 -0.07631029  1.2049200 -0.1112567   0.1591755
## 2 -0.2953914 -0.5016290  0.36107795 -0.5096937  0.1180883  -0.7107373
## 3  0.0985208 -0.1817299 -0.70455885 -0.2023993 -0.1362761   1.3770536
##   gethitched
## 1 -0.1390230
## 2 -0.1390230
## 3  0.4549845
```

This result shows:

- *Cluster 1*: divyear = mostly 90, momint = very close, dadint = not close, livewithmom = mostly mother, depression = not often, (gethiched) marry = will likely not get married. Cluster 1 represents mostly adolescents that are closer to mom than dad. These young adults do not often feel depressed and they may

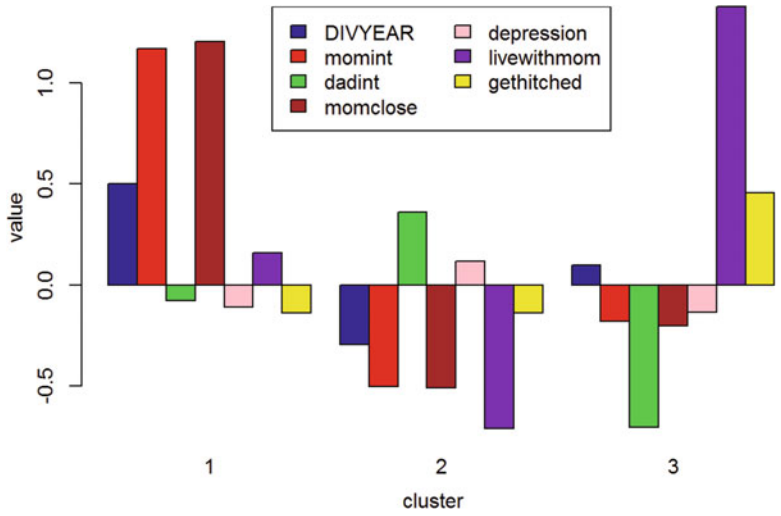

Fig. 13.6 Barplot illustrating the features discriminating between the three cohorts in the divorce sonsequences on young adults dataset

avoid getting married. These young adults tends to be not be too emotional and do not value family.

- *Cluster 2*: divyear = mostly 89, momint = not close, dadint = very close, livewithmom = father, depression = mild, marry = do not know/not inclined. Cluster 2 includes children that mostly live with dad and only feel close to dad. These people don't felt severely depressed and are not inclined to marry. These young adults may prefer freedom and tend to be more naive.
- *Cluster 3*: divyear = mix of 89 and 90, momint = not close, dadint = not at all, livewithmom = mother, depression = sometimes, marry = tend to get married. Cluster 3 contains children that did not feel close to either dad or mom. They sometimes felt depressed and are willing to build their own family. These young adults seem to be more independent.

We can see that these three different clusters do contain three alternative types of young adults. Bar plots provide an alternative strategy to visualize the difference between clusters (Fig. 13.6).

```
par(mfrow=c(1, 1), mar=c(4, 4, 4, 2))
myColors <- c("darkblue","red","green","brown","pink","purple","yellow")
barplot(t(diz_clussters$centers), beside = TRUE, xlab="cluster",
ylab="value", col = myColors)
legend("topleft", ncol=2, legend = c("DIVYEAR", "momint", "dadint",
"momclose", "depression", "livewithmom", "gethitched"), fill = myColors)
```

For each of the three clusters, the bars in the plot above represent the following order of features `DIVYEAR, momint, dadint, momclose, depression, livewithmom, gethitched`.

13.4.5 Step 5: Usage of Cluster Information

Clustering results could be utilized as new information augmenting the original dataset. For instance, we can add a *cluster* label in our `divorce` dataset:

```
divorce$clusters<-diz_clussters$cluster
divorce[1:5, ]

##   DIVYEAR momint dadint momclose depression livewithmom gethitched
## 1       1      3      2        2          3           1          3
## 2       1      3      2        3          2           2          2
## 3       0      1      3        1          4           1          2
## 4       1      3      4        3          3           1          3
## 5       0      1      3        2          1           1          2
##   clusters
## 1        1
## 2        1
## 3        2
## 4        1
## 5        2
```

We can also examine the relationship between live with mom and feel close to mom by displaying a scatter plot of these two variables. If we suspect that young adults' personality might affect this relationship, then we could consider the potential personality (cluster type) in the plot. The cluster labels associated with each participant are printed in different positions relative to each pair of observations, `(livewithmom, momint)` (Fig. 13.7).

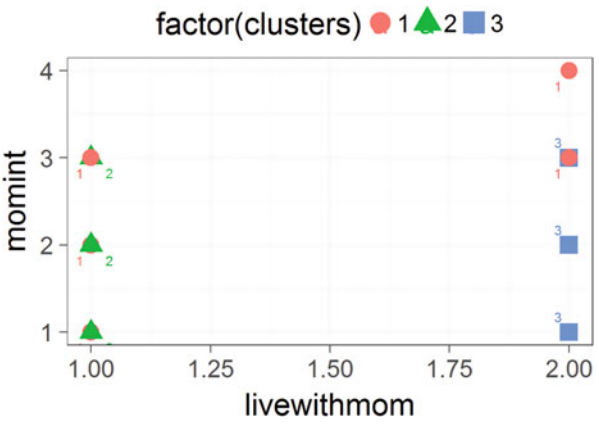

Fig. 13.7 Drill down for one feature (leave-with-mom) between the three cohorts

```
require(ggplot2)
ggplot(divorce, aes(livewithmom, momint), main="Scatterplot Live with mom vs
feel close to mom") +
  geom_point(aes(colour = factor(clusters), shape=factor(clusters), stroke =
8), alpha=1) +
  theme_bw(base_size=25) +
  geom_text(aes(label=ifelse(clusters%in%1, as.character(clusters), ''), hju
st=2, vjust=2, colour = factor(clusters)))+
  geom_text(aes(label=ifelse(clusters%in%2, as.character(clusters), ''), hju
st=-2, vjust=2, colour = factor(clusters)))+
  geom_text(aes(label=ifelse(clusters%in%3, as.character(clusters), ''), hju
st=2, vjust=-1, colour = factor(clusters))) +
  guides(colour = guide_legend(override.aes = list(size=8))) +
theme(legend.position="top")
```

We used `ggplot()` function in `ggplot2` package to label points with cluster types. `ggplot(divorce, aes(livewithmom, momint)) + geom_point()` gives us the scatterplot, and the three `geom_text()` functions help us label the points with the corresponding cluster identifiers.

This picture shows that live with mom does not necessarily mean young adults will feel close to mom. For "emotional" (Cluster 1) young adults, they felt close to their mom whether they live with their mom or not. "Naive" (Cluster 2) young adults feel closer to mom if they live with mom. However, they tend to be estranged from their mother. "Independent" (Cluster 3) young adults are opposite to Cluster 1. They felt closer to mom if they don't live with her.

13.5 Model Improvement

Let's still use the divorce data to illustrate a model improvement using **k-means++**. (Appropriate) initialization of the **k-means** algorithm is of paramount importance. The **k-means++** extension provides a practical strategy to obtain an optimal initialization for k-means clustering using a predefined `kpp_init` method.

```
# install.packages("matrixStats")
require(matrixStats)

kpp_init = function(dat, K) {
  x = as.matrix(dat)
  n = nrow(x)
  # Randomly choose a first center
  centers = matrix(NA, nrow=K, ncol=ncol(x))
  set.seed(123)
  centers[1,] = as.matrix(x[sample(1:n, 1),])
  for (k in 2:K) {
    # Calculate dist^2 to closest center for each point
    dists = matrix(NA, nrow=n, ncol=k-1)
    for (j in 1:(k-1)) {
      temp = sweep(x, 2, centers[j,], '-')
      dists[,j] = rowSums(temp^2)
    }
    dists = rowMins(dists)
```

```
    # Draw next center with probability proportional to dist^2
    cumdists = cumsum(dists)
    prop = runif(1, min=0, max=cumdists[n])
    centers[k,] = as.matrix(x[min(which(cumdists > prop)),])
  }
  return(centers)
}

clust_kpp = kmeans(di_z, kpp_init(di_z, 3), iter.max=100, algorithm='Lloyd')
```

We can observe some differences.

```
clust_kpp$centers

##      DIVYEAR     momint     dadint   momclose depression livewithmom
## 1  0.3741445  1.2578161 -0.6636602  0.5610651 -0.1505730  -0.4124815
## 2 -0.2659149 -0.5798266  0.3805174 -0.2538624  0.1639572  -0.5560862
## 3  0.3508225  0.5269697 -0.4329499  0.2251408 -0.2594488   1.3770536
##   gethitched
## 1  0.9990071
## 2 -0.1489684
## 3 -0.2285310
```

This improvement is not substantial; the new overall average Silhouette value remains 0.2 for **k-means++**. Third compares to the value of 0.2 reported for the earlier k-means clustering, albeit the three groups generated by each method are quite distinct. In addition, the number of "mis-clustered" instances remains 2 although their Silhouette values are rather smaller than before, and the overall Cluster 1 Silhouette average value is low (0.006) (Fig. 13.8).

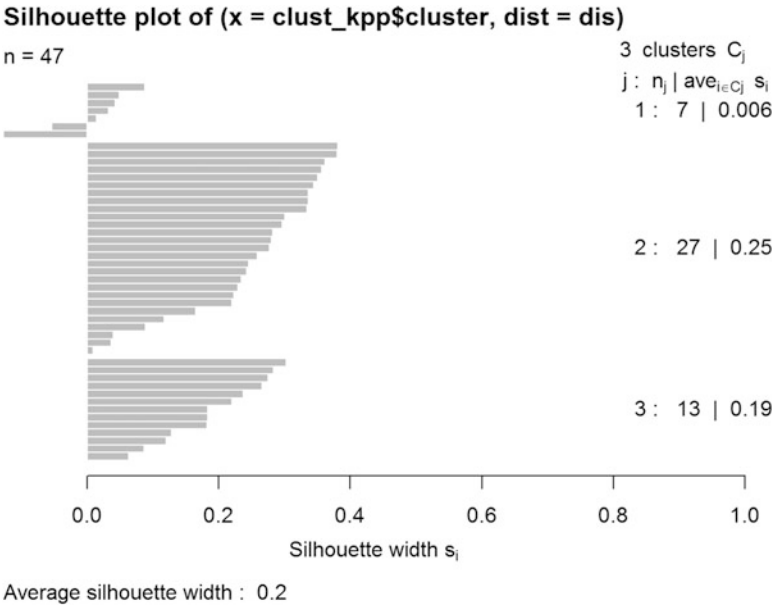

Fig. 13.8 Silhouette plot for k-means++ classification

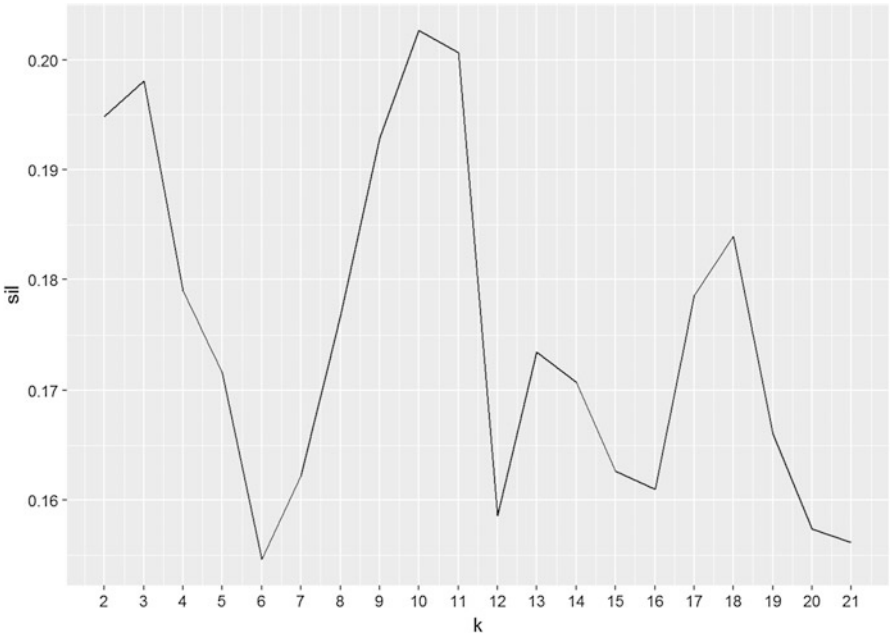

Fig. 13.9 Evolution of the average silhouette value with respect to the number of clusters

```
sil2 = silhouette(clust_kpp$cluster, dis)
summary(sil2)

## Silhouette of 47 units in 3 clusters from silhouette.default(x = clust_kp
p$cluster, dist = dis) :
##  Cluster sizes and average silhouette widths:
##          7         27         13
## 0.00644352 0.24933847 0.19476785
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.12750  0.08781  0.22950  0.19810  0.29050  0.38120

plot(sil2)
```

13.5.1 *Tuning the Parameter* k

Similar to what we performed for KNN and SVM, we can tune the **k-means** parameters, including centers initialization and *k* (Fig. 13.9).

```
n_rows <- 21
mat = matrix(0,nrow = n_rows)
for (i in 2:n_rows){
  set.seed(321)
  clust_kpp = kmeans(di_z, kpp_init(di_z, i), iter.max=100, algorithm='Lloyd
')
  sil = silhouette(clust_kpp$cluster, dis)
  mat[i] = mean(as.matrix(sil)[,3])
}
colnames(mat) <- c("Avg_Silhouette_Value")
mat
```

```
##       Avg_Silhouette_Value
##  [1,]            0.0000000
##  [2,]            0.1948335
##  [3,]            0.1980686
##  [4,]            0.1789654
##  [5,]            0.1716270
##  [6,]            0.1546357
##  [7,]            0.1622488
##  [8,]            0.1767659
##  [9,]            0.1928883
## [10,]            0.2026559
## [11,]            0.2006313
## [12,]            0.1586044
## [13,]            0.1735035
## [14,]            0.1707446
## [15,]            0.1626367
## [16,]            0.1609723
## [17,]            0.1785733
## [18,]            0.1839546
## [19,]            0.1660019
## [20,]            0.1573574
## [21,]            0.1561791
```

```
ggplot(data.frame(k=2:n_rows,sil=mat[2:n_rows]),aes(x=k,y=sil))+
  geom_line()+
  scale_x_continuous(breaks = 2:n_rows)
```

This suggests that $k \sim 3$ may be an appropriate number of clusters to use in this case.

Next, let's set the maximal iteration of the algorithm and rerun the model with optimal $\mathtt{k} = 2$, $\mathtt{k} = 3$ or $\mathtt{k} = 10$. Below, we just demonstrate the results for $k = 3$. There are still 2 mis-clustered observations, which is not a significant improvement on the prior model according to the average Silhouette measure (Fig. 13.10).

```
k <- 3
set.seed(31)
clust_kpp = kmeans(di_z, kpp_init(di_z, k), iter.max=200, algorithm="MacQuee
n")
sil3 = silhouette(clust_kpp$cluster, dis)
summary(sil3)
```

```
## Silhouette of 47 units in 3 clusters from silhouette.default(x = clust_kp
p$cluster, dist = dis) :
##  Cluster sizes and average silhouette widths:
##         10         22         15
## 0.02096194 0.30414984 0.15474729
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1365  0.1032  0.1971  0.1962  0.3122  0.4113
```

```
plot(sil3)
```

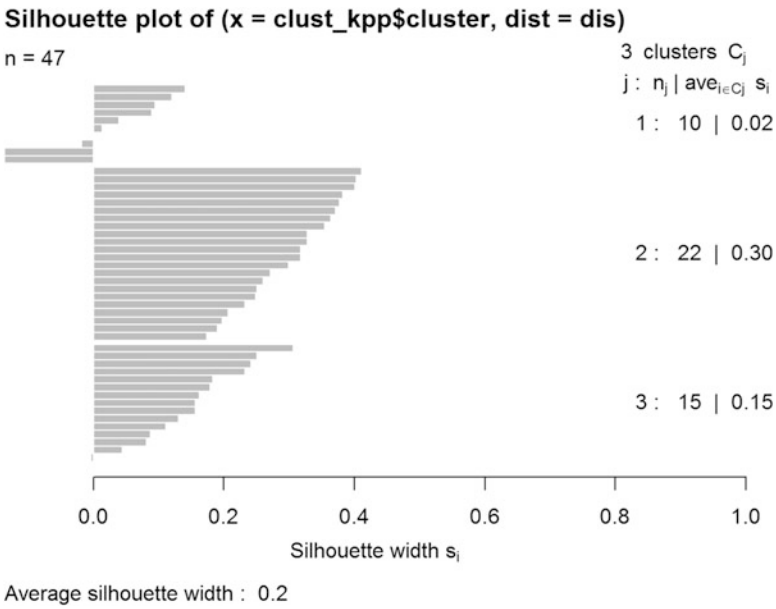

Fig. 13.10 Silhouette plot for the optimal $k = 3$ andd `kpp_init` Initialization

Note that we now see 3 cases of group 1 that have negative silhouette values (previously we had only 2), albeit the overall average silhouette remains 0.2.

13.6 Case Study 2: Pediatric Trauma

Let's go through another example demonstrating the *k-means* clustering method using a larger dataset.

13.6.1 Step 1: Collecting Data

The dataset we will interrogate now includes Services Utilization by Trauma-Exposed Children in the US data, which is located in our case-studies folder. This case study examines associations between post-traumatic psychopathology and service utilization by trauma-exposed children.

Variables:

- **id**: Case identification number.
- **sex**: Female or male, dichotomous variable (1 = female, 0 = male).
- **age**: Age of child at time of seeking treatment services. Interval-level variable, score range = 0–18.
- **race**: Race of child seeking treatment services. Polytomous variable with 4 categories (1 = black, 2 = white, 3 = hispanic, 4 = other).

- **cmt**: The child was exposed to child maltreatment trauma - dichotomous variable (1 = yes, 0 = no).
- **traumatype**: Type of trauma exposure the child is seeking treatment sore. Polytomous variable with 5 categories ("sexabuse" = sexual abuse, "physabuse" = physical abuse, "neglect" = neglect, "psychabuse" = psychological or emotional abuse, "dvexp" = exposure to domestic violence or intimate partner violence).
- **ptsd**: The child has current post-traumatic stress disorder. Dichotomous variable (1 = yes, 0 = no).
- **dissoc**: The child currently has a dissociative disorder (PTSD dissociative subtype, DESNOS, DDNOS). Interval-level variable, score range = 0–11.
- **service**: Number of services the child has utilized in the past 6 months, including primary care, emergency room, outpatient therapy, outpatient psychiatrist, inpatient admission, case management, in-home counseling, group home, foster care, treatment foster care, therapeutic recreation or mentor, department of social services, residential treatment center, school counselor, special classes or school, detention center or jail, probation officer. Interval-level variable, score range = 0–19.
- **Note**: These data (`Case_04_ChildTrauma._Data.csv`) are tab-delimited.

13.6.2 Step 2: Exploring and Preparing the Data

First, we need to load the dataset into R and report its summary and dimensions.

```
trauma<-read.csv("https://umich.instructure.com/files/399129/download?downlo
ad_frd=1", sep = " ")
summary(trauma); dim(trauma)
```

```
##        id              sex              age              ses
##  Min.   :   1.0   Min.   :0.000   Min.   : 2.000   Min.   :0.00
##  1st Qu.: 250.8   1st Qu.:0.000   1st Qu.: 7.000   1st Qu.:0.00
##  Median : 500.5   Median :1.000   Median : 9.000   Median :0.00
##  Mean   : 500.5   Mean   :0.506   Mean   : 8.982   Mean   :0.18
##  3rd Qu.: 750.2   3rd Qu.:1.000   3rd Qu.:11.000   3rd Qu.:0.00
##  Max.   :1000.0   Max.   :1.000   Max.   :25.000   Max.   :1.00
##        race          traumatype       ptsd          dissoc
##  black   :200   dvexp     :250   Min.   :0.00   Min.   :0.000
##  hispanic:100   neglect   :350   1st Qu.:0.00   1st Qu.:0.000
##  other   :100   physabuse :100   Median :0.00   Median :1.000
##  white   :600   psychabuse:200   Mean   :0.29   Mean   :0.598
##                 sexabuse  :100   3rd Qu.:1.00   3rd Qu.:1.000
##                                  Max.   :1.00   Max.   :1.000
##     service
##  Min.   : 0.000
##  1st Qu.: 8.000
##  Median :10.000
##  Mean   : 9.926
##  3rd Qu.:12.000
##  Max.   :20.000

## [1] 1000    9
```

In the summary we see two factors `race` and `traumatype`. `Traumatype` codes the real classes we are interested in. If the clusters created by the model are quite similar to the trauma types, our model may have a quite reasonable interpretation. Let's also create a dummy variable for each racial category.

```
trauma$black<-ifelse(trauma$race=="black", 1, 0)
trauma$hispanic<-ifelse(trauma$race=="hispanic", 1, 0)
trauma$other<-ifelse(trauma$race=="other", 1, 0)
trauma$white<-ifelse(trauma$race=="white", 1, 0)
```

Then, we will remove `traumatype` the class variable from the dataset to avoid biasing the clustering algorithm. Thus, we are simulating a real biomedical case-study where we do not necessarily have the actual class information available, i.e., classes are latent features.

```
trauma_notype<-trauma[, -c(1, 5, 6)]
```

13.6.3 Step 3: Training a Model on the Data

Similar to case-study 1, let's standardize the dataset and fit a k-means model.

```
tr_z<- as.data.frame(lapply(trauma_notype, scale))
str(tr_z)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ sex     : num  0.988 0.988 -1.012 -1.012 0.988 ...
##  $ age     : num  -0.997 1.677 -0.997 0.674 -0.662 ...
##  $ ses     : num  -0.468 -0.468 -0.468 -0.468 -0.468 ...
##  $ ptsd    : num  1.564 -0.639 -0.639 -0.639 1.564 ...
##  $ dissoc  : num  0.819 -1.219 0.819 0.819 0.819 ...
##  $ service : num  2.314 0.678 -0.303 0.351 1.66 ...
##  $ black   : num  2 2 2 2 2 ...
##  $ hispanic: num  -0.333 -0.333 -0.333 -0.333 -0.333 ...
##  $ other   : num  -0.333 -0.333 -0.333 -0.333 -0.333 ...
##  $ white   : num  -1.22 -1.22 -1.22 -1.22 -1.22 ...
```

```
set.seed(1234)
trauma_clusters<-kmeans(tr_z, 6)
```

Here we use $k = 6$ in the hope that we may have 5 of these clusters match the specific 5 trauma types. In this case study, we have 1000 observations and $k = 6$ may be a reasonable option.

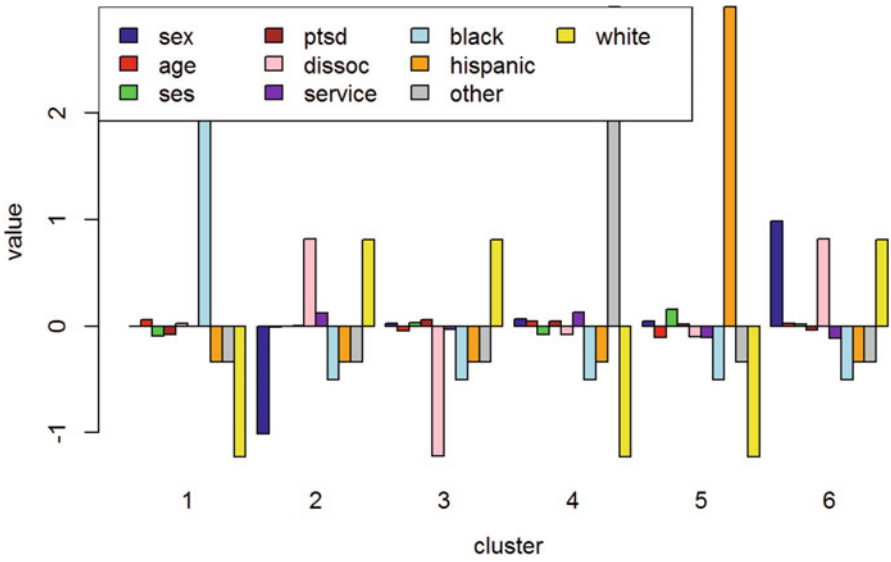

Fig. 13.11 Key predictors discriminating between the 6 cohorts in the trauma study

13.6.4 Step 4: Evaluating Model Performance

To assess the clustering model results, we can examine the resulting clusters (Fig. 13.11).

```
trauma_clusters$centers

##            sex          age          ses         ptsd      dissoc
## 1 -0.001999144  0.061154336 -0.091055799 -0.077094361  0.02446247
## 2 -1.011566709 -0.006361734 -0.000275301  0.002214351  0.81949287
## 3  0.026286613 -0.043755817  0.029890657  0.064206246 -1.21904661
## 4  0.067970886  0.046116384 -0.078047828  0.044053921 -0.07746450
## 5  0.047979449 -0.104263129  0.156095655  0.022026960 -0.09784989
## 6  0.987576985  0.028799955  0.019511957 -0.038046568  0.81949287
##        service      black   hispanic      other      white
## 1  0.001308569  1.9989997 -0.3331666 -0.3331666 -1.2241323
## 2  0.126332303 -0.4997499 -0.3331666 -0.3331666  0.8160882
## 3 -0.030083167 -0.4997499 -0.3331666 -0.3331666  0.8160882
## 4  0.128894052 -0.4997499 -0.3331666  2.9984996 -1.2241323
## 5 -0.103376956 -0.4997499  2.9984996 -0.3331666 -1.2241323
## 6 -0.111481162 -0.4997499 -0.3331666 -0.3331666  0.8160882
myColors <- c("darkblue", "red", "green", "brown", "pink", "purple", "lightb
lue", "orange", "grey", "yellow")
barplot(t(trauma_clusters$centers), beside = TRUE, xlab="cluster",
ylab="value", col = myColors)
legend("topleft", ncol=4, legend = c("sex", "age", "ses", "ptsd", "dissoc",
"service", "black", "hispanic", "other", "white"), fill = myColors)
```

On this barplot, the bars in each cluster represents `sex`, `age`, `ses`, `ptsd`, `dissoc`, `service`, `black`, `hispanic`, `other`, and `white`, respectively. It is quite obvious that each cluster has some unique features.

Next, we can compare the *k-means* computed cluster labels to the *original labels*. Let's evaluate the similarities between the automated cluster labels and their real class counterparts using a confusion matrix table, where rows represent the k-means clusters, columns show the actual labels, and the cell values include the frequencies of the corresponding pairings.

```
trauma$clusters<-trauma_clusters$cluster
table(trauma$clusters, trauma$traumatype)

##
##     dvexp neglect physabuse psychabuse sexabuse
##   1     0       0       100          0      100
##   2    10     118         0         61        0
##   3    23     133         0         79        0
##   4   100       0         0          0        0
##   5   100       0         0          0        0
##   6    17      99         0         60        0
```

We can see that all of the children in Cluster 4 belong to `dvexp` (exposure to domestic violence or intimate partner violence). If we use the mode of each cluster to be the class for that group of children, we can classify 63 `sexabuse` cases, 279 `neglect` cases, 41 `physabuse` cases, 100 `dvexp` cases, and another 71 `neglect` cases. That is 554 cases out of 1,000 cases identified with correct class. The model has a problem in distinguishing between `neglect` and `psychabuse`, but it has a good accuracy.

Let's review the output Silhouette value summary. It works well as only a small portion of samples appear mis-clustered.

```
dis_tra = dist(tr_z)
sil_tra = silhouette(trauma_clusters$cluster, dis_tra)
summary(sil_tra)

## Silhouette of 1000 units in 6 clusters from silhouette.default(x = trauma
_clusters$cluster, dist = dis_tra) :
##  Cluster sizes and average silhouette widths:
##       200       189       235       100       100       176
## 0.2595725 0.2185706 0.1039559 0.3223076 0.3199830 0.2423110
## Individual silhouette widths:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.008893 0.139100 0.234400 0.224500 0.303300 0.388200

#plot(sil_tra)
# report the overall mean silhouette value
mean(sil_tra[,"sil_width"])

## [1] 0.2245298

# The sil object colnames  are ("cluster", "neighbor", "sil_width")
```

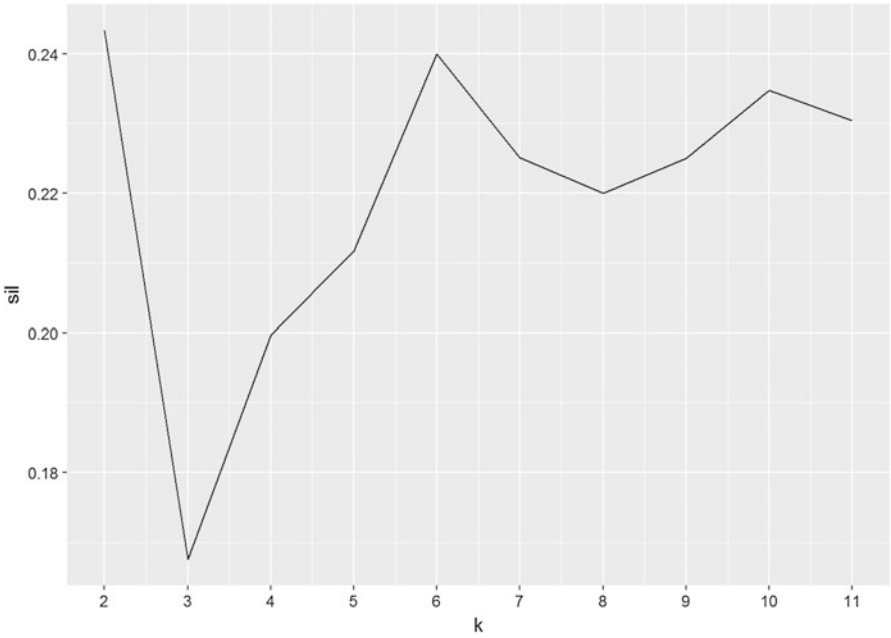

Fig. 13.12 Evolution of the average silhouette value with respect to the number of clusters

Next, let's try to tune k with **k-means++** and see if $k = 6$ appears to be optimal (Fig. 13.12).

```
mat = matrix(0,nrow = 11)
for (i in 2:11){
  set.seed(321)
  clust_kpp = kmeans(tr_z, kpp_init(tr_z, i), iter.max=100, algorithm='Lloyd
')
  sil = silhouette(clust_kpp$cluster, dis_tra)
  mat[i] = mean(as.matrix(sil)[,3])
}
mat

##             [,1]
##  [1,] 0.0000000
##  [2,] 0.2433222
##  [3,] 0.1675486
##  [4,] 0.1997315
##  [5,] 0.2116534
##  [6,] 0.2400086
##  [7,] 0.2251367
##  [8,] 0.2199859
##  [9,] 0.2249569
## [10,] 0.2347122
## [11,] 0.2304451

ggplot(data.frame(k=2:11,sil=mat[2:11]),aes(x=k,y=sil))+geom_line()+scale_x_
continuous(breaks = 2:11)
```

Finally, let's use **k-means++** with $k = 6$ and set the algorithm's maximal iteration before rerunning the experiment:

```
set.seed(1234)
clust_kpp = kmeans(tr_z, kpp_init(tr_z, 6), iter.max=100, algorithm='Lloyd')
sil = silhouette(clust_kpp$cluster, dis_tra)
summary(sil)
```

```
## Silhouette of 1000 units in 6 clusters from silhouette.default(x = clust_
kpp$cluster, dist = dis_tra) :
##  Cluster sizes and average silhouette widths:
##       422       100       178        85        15       200
## 0.2166778 0.3353976 0.1898492 0.2478090 0.2294502 0.2836607
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03672 0.19730 0.23080 0.24000 0.27710 0.40650
```

```
# plot(sil)
# report the overall mean silhouette value
mean(sil[,"sil_width"])
```

```
## [1] 0.2400086
```

13.6.5 Practice Problem: Youth Development

Use the Boys Town Study of Youth Development data, second case study, CaseStudy02_Boystown_Data.csv, which we used in Chap. 7, to find clusters using variables like GPA, alcohol abuse, attitudes on drinking, social status, parent closeness, and delinquency for clustering (all variables other than gender and ID).

First, we must load the data and transfer `sex`, `dadjob`, and `momjob` into dummy variables.

```
boystown<-read.csv("https://umich.instructure.com/files/399119/download?down
load_frd=1", sep=" ")
boystown$sex<-boystown$sex-1
boystown$dadjob <- (-1)*(boystown$dadjob-2)
boystown$momjob <- (-1)*(boystown$momjob-2)
str(boystown)
```

```
## 'data.frame':    200 obs. of  11 variables:
##  $ id        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ sex       : num  0 0 0 0 1 1 0 0 1 1 ...
##  $ gpa       : int  5 0 3 2 3 3 1 5 1 3 ...
##  $ Alcoholuse: int  2 4 2 2 6 3 2 6 5 2 ...
##  $ alcatt    : int  3 2 3 1 2 0 0 3 0 1 ...
##  $ dadjob    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ momjob    : num  0 0 0 0 1 0 0 0 1 1 ...
##  $ dadclose  : int  1 3 2 1 2 1 3 6 3 1 ...
##  $ momclose  : int  1 4 2 2 1 2 1 2 3 2 ...
##  $ larceny   : int  1 0 0 3 1 0 0 0 1 1 ...
##  $ vandalism : int  3 0 2 2 2 0 5 1 4 0 ...
```

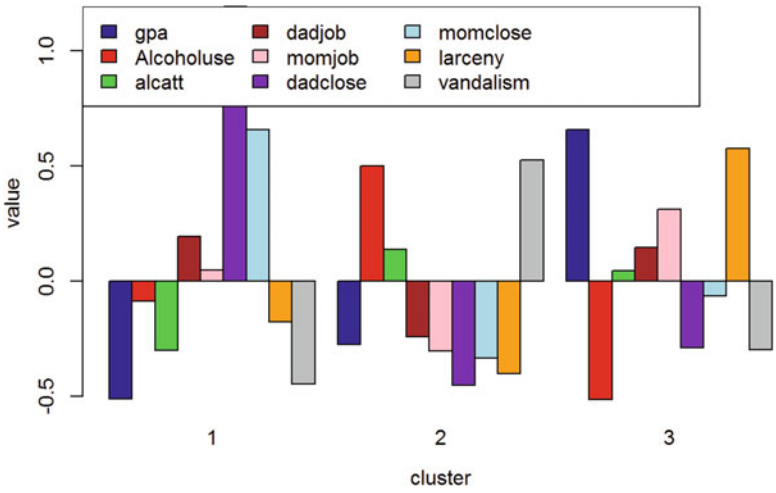

Fig. 13.13 Main features discriminating between the 3 cohorts in the divorce impact on youth study

Then, extract all the variables, except the first two columns (subject identifiers and genders).

```
boystown_sub<-boystown[, -c(1, 2)]
```

Next, we need to standardize and clustering the data with $k = 3$. You may have the following centers (numbers could be a little different) (Fig. 13.13).

```
##          gpa  Alcoholuse      alcatt     dadjob      momjob   dadclose
## 1 -0.5101243 -0.08555163 -0.30098866  0.1939577  0.04868109  1.1914502
## 2 -0.2753631  0.49998217  0.13804858 -0.2421906 -0.30151766 -0.4521484
## 3  0.6590193 -0.51256447  0.04599325  0.1451756  0.31107377 -0.2896562
##      momclose    larceny  vandalism
## 1  0.65647213 -0.1755012 -0.4453044
## 2 -0.33341358 -0.4017282  0.5252308
## 3 -0.06343891  0.5769583 -0.2981561
```

Add *k-means* cluster labels as a new (last) column back in the original dataset.

To investigate the gender distribution within different clusters we may use `aggregate()`.

```
# Compute the averages for the variable 'sex', grouped by cluster
aggregate(data=boystown, sex~clusters, mean)
```

```
##   clusters       sex
## 1        1 0.6875000
## 2        2 0.5802469
## 3        3 0.6760563
```

Here `clusters` is the new vector indicating cluster labels. The gender distribution does not vary much between different cluster labels (Fig. 13.14).

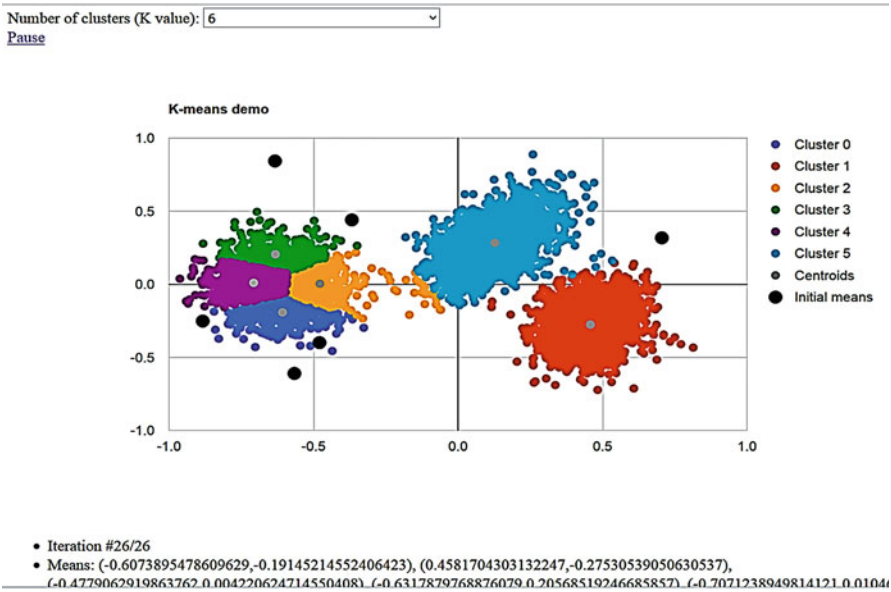

Fig. 13.14 Live demo: k-means point clustering

This k-means live demo shows point clustering (Applies Multiclass AdaBoost.M1, SAMME and Bagging algorithm) http://olalonde.github.com/kmeans.js.

13.7 Hierarchical Clustering

There are a number of `R` **hierarchical clustering** packages, including:

- `hclust` in base R.
- `agnes` in the `cluster` package.

Alternative distance measures (or linkages) can be used in all Hierarchical Clustering, e.g., *single*, *complete* and *ward*.

We will demonstrate hierarchical clustering using case-study 1 (Divorce and Consequences on Young Adults). Pre-set $k = 3$ and notice that we have to use normalized data for hierarchical clustering.

```
require(cluster)
pitch_sing = agnes(di_z, diss=FALSE, method='single')
pitch_comp = agnes(di_z, diss=FALSE, method='complete')
pitch_ward = agnes(di_z, diss=FALSE, method='ward')
sil_sing = silhouette(cutree(pitch_sing, k=3), dis)
sil_comp = silhouette(cutree(pitch_comp, k=3), dis)
# try 10 clusters, see plot above
sil_ward = silhouette(cutree(pitch_ward, k=10), dis)
```

You can generate the hierarchical plot by `ggdendrogram` in the package `ggdendro` (Figs. 13.15 and 13.16).

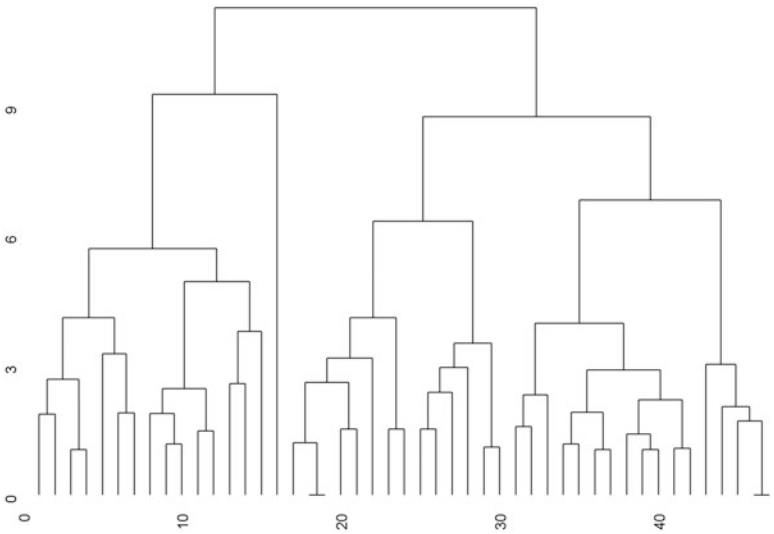

Fig. 13.15 Hierarchical clustering using the Ward method

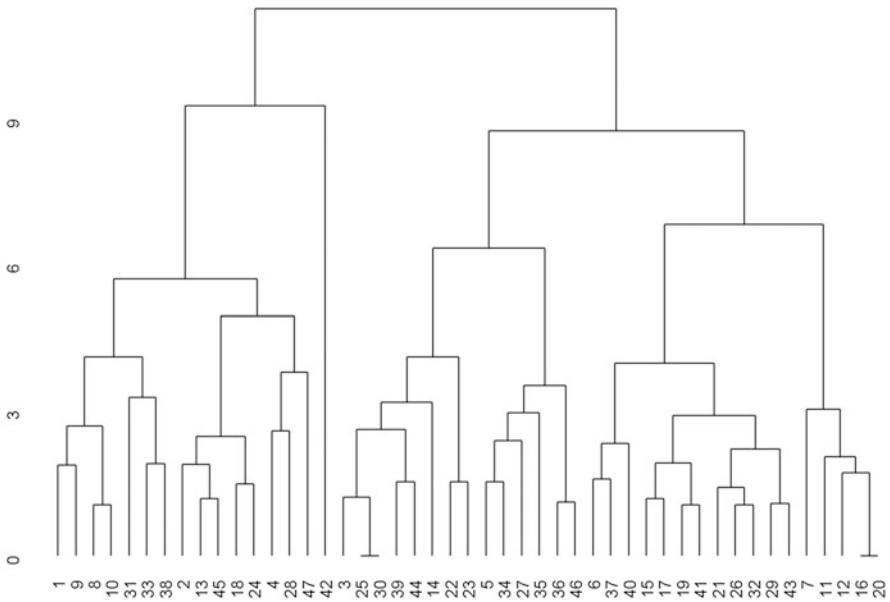

Fig. 13.16 Ten-level hierarchical clustering using the Ward method

```
# install.packages("ggdendro")
require(ggdendro)

ggdendrogram(as.dendrogram(pitch_ward), leaf_labels=FALSE, labels=FALSE)
```

Fig. 13.17 Silhouette plot for hierarchical clustering using the Ward method

```
mean(sil_ward[,"sil_width"])

## [1] 0.2398738

ggdendrogram(as.dendrogram(pitch_ward), leaf_labels=TRUE, labels=T, size=10)
```

Generally speaking, the best result should come from **wald** linkage, but you should also try complete linkage (method = 'complete'). We can see that the hierarchical clustering result (average silhouette value ~0.24) mostly agrees with the prior *k-means* (0.2) and *k-means++* (0.2) results (Fig. 13.17).

```
summary(sil_ward)

## Silhouette of 47 units in 10 clusters from silhouette.default(x = cutree(
pitch_ward, k = 10), dist = dis) :
##  Cluster sizes and average silhouette widths:
##           4           5           6           3           6          12
##  0.25905454  0.29195989  0.29305926 -0.02079056  0.19263836  0.26268274
##           5           2           3           1
##  0.32594365  0.44074717  0.08760990  0.00000000
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1477  0.1231  0.2577  0.2399  0.3524  0.5176

plot(sil_ward)
```

13.8 Gaussian Mixture Models

More details about Gaussian mixture models (GMM) are provided in the supporting materials online. Below is a brief introduction to GMM using the `Mclust` function in the *R* package `mclust`.

For multivariate mixture, there are totally 14 possible models:

- "EII" = spherical, equal volume
- "VII" = spherical, unequal volume
- "EEI" = diagonal, equal volume and shape
- "VEI" = diagonal, varying volume, equal shape
- "EVI" = diagonal, equal volume, varying shape
- "VVI" = diagonal, varying volume and shape
- "EEE" = ellipsoidal, equal volume, shape, and orientation
- "EVE" = ellipsoidal, equal volume and orientation (*)
- "VEE" = ellipsoidal, equal shape and orientation (*)
- "VVE" = ellipsoidal, equal orientation (*)
- "EEV" = ellipsoidal, equal volume and equal shape
- "VEV" = ellipsoidal, equal shape
- "EVV" = ellipsoidal, equal volume (*)
- "VVV" = ellipsoidal, varying volume, shape, and orientation

For more practical details, you may refer to Mclust. For more theoretical details, see C. Fraley and A. E. Raftery (2002).

Let's use the Divorce and Consequences on Young Adults dataset for a demonstration.

```
library(mclust)

set.seed(1234)
gmm_clust = Mclust(di_z)
gmm_clust$modelName

## [1] "EEE"
```

Thus, the optimal model here is "EEE" (Figs. 13.18, 13.19, and 13.20).

```
plot(gmm_clust$BIC, legendArgs = list(x = "bottom", ncol = 2, cex = 1))
```

```
plot(gmm_clust,what = "density")
```

```
plot(gmm_clust,what = "classification")
```

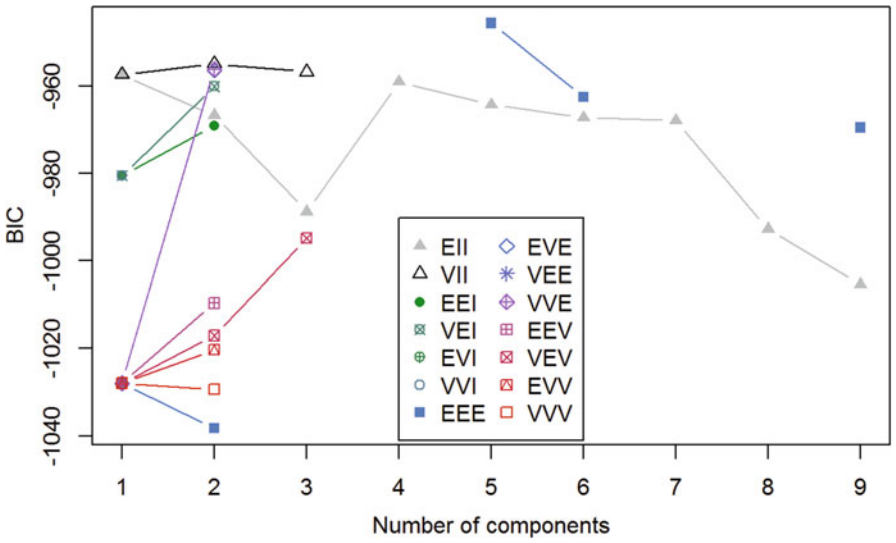

Fig. 13.18 Bayesian information criterion plots for different GMM classification models for the divorce youth data

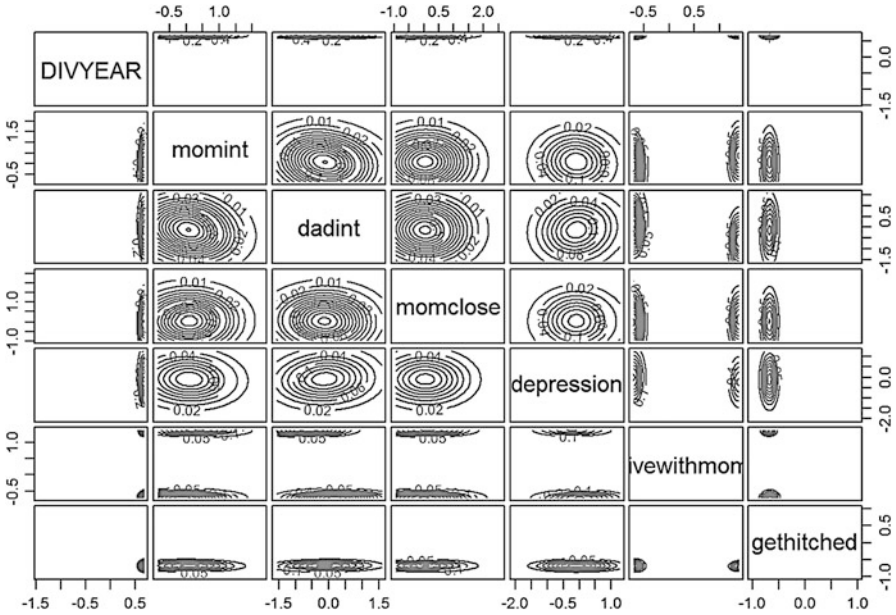

Fig. 13.19 Pairs plot of the GMM clustering density

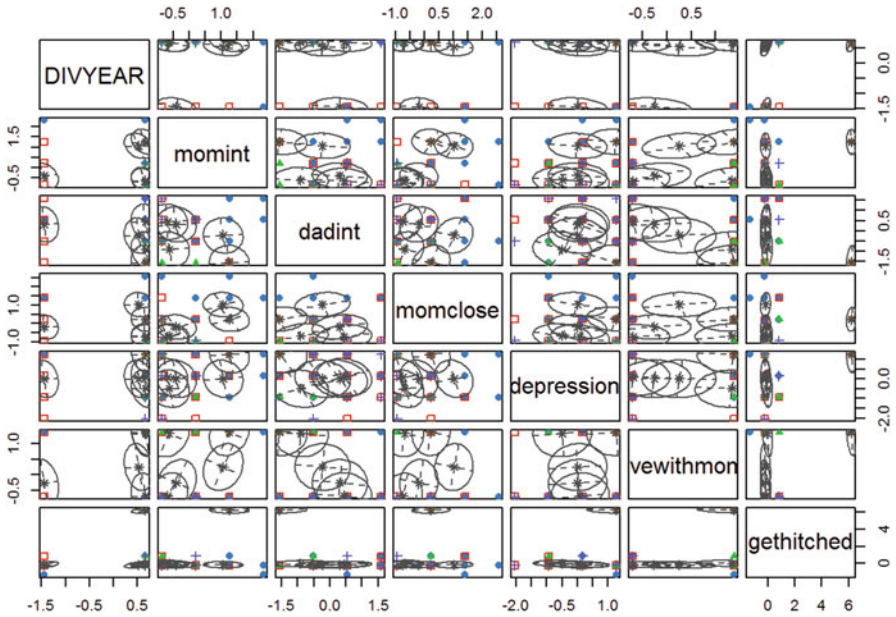

Fig. 13.20 Pairs plot of the GMM classification results

13.9 Summary

- k-means clustering may be most appropriate for exploratory data analytics. It is highly flexible and fairly efficient in terms of tessellating data into groups.
- It can be used for data that has no *Apriori* classes (labels).
- Generated clusters may lead to phenotype stratification and/or be compared against known clinical traits.

Try to use these techniques with other data from the list of our Case-Studies.

13.10 Assignments: 13. k-Means Clustering

Use the Amyotrophic Lateral Sclerosis (ALS) dataset. This case-study examines the patterns, symmetries, associations and causality in a rare but devastating disease, amyotrophic lateral sclerosis (ALS). A major clinically relevant question in this biomedical study is: *What patient phenotypes can be automatically and reliably identified and used to predict the change of the ALSFRS slope over time?*. This problem aims to explore the data set by unsupervised learning.

- Load and prepare the data.
- Perform summary and preliminary visualization.

- Train a **k-means** model on the data, select *k*
- as we mentioned in Chap. 13.
- Evaluate the model performance and report the center of clusters and silhouette plots. Explain details (Note: Since we have 100 dimensions, it may be difficult to use bar plots, so show the centers only).
- Tune parameters and plot with **k-means++**.
- Rerun the model with optimal parameters and interpret the clustering results.
- Apply *Hierarchical Clustering* on three different linkages and compare the corresponding **Silhouette** plots.
- Fit a Gaussian mixture model, select the optimal model and draw **BIC** and **Silhouette** plots. (Hint, you need to sample part of data or it could be very time consuming).
- Compare the result of the above methods.

References

Wu, J. (2012) *Advances in K-means Clustering: A Data Mining Thinking*, Springer Science & Business Media, ISBN 3642298079, 9783642298073.

Dinov, ID. (2008) Expectation Maximization and Mixture Modeling Tutorial. Statistics Online Computational Resource. UCLA: Statistics Online Computational Resource. Retrieved from: http://escholarship.org/uc/item/1rb70972.

Celebi, ME (ed.) (2014) *Partitional Clustering Algorithms*, SpringerLink: Bücher, ISBN 3319092596, 9783319092591.

Fraley, C and Raftery, AE. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association, 97,* 611–631.