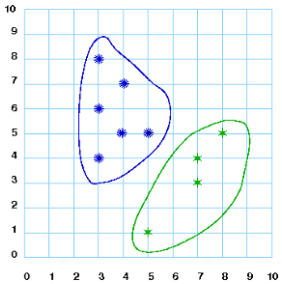
Chapter 9

Cluster Analysis



The clustering technique is one of the core tools that is used by the data miner. Clustering gives us the opportunity to group observations in a generally unguided fashion according to how similar they are. This is done on the basis of a measure of the distance between observations. For example, we might have a dataset that is made up of school children of various heights, a range of weights, and different ages. Depending on what is needed to solve the problem at hand, we might wish to group the students into smaller, more definable groups and then compare different variables common to all groupings. Each group may have different ranges, minimums and maximums, and so on that represent that group. Clustering allows the data miner to break data into more meaningful groups and then contrast the different clusters against each other. Clusters can also be useful in grouping observations to help make the smaller datasets easier to manage. The aim of clustering is often to identify groups of observations that are close together but as a group are quite separate from other groups.

Numerous algorithms have been developed for clustering. In this chapter, we focus primarily on the k-means clustering algorithm. The algorithm will identify a collection of k clusters using a heuristic search starting with a selection of k randomly chosen clusters.

G. Williams, *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery*, Use R, DOI 10.1007/978-1-4419-9890-3_9, © Springer Science+Business Media, LLC 2011

9.1 Knowledge Representation

A model built using the k-means algorithm represents the clusters as a collection of k means. The observations in the dataset are associated with their closest "mean" and thus are partitioned into k clusters. The *mean* of a particular *numeric* variable for a collection of observations is the average value of that variable over those observations. The *means* for the collection of observations that form one of the k clusters in any particular clustering are then the collection of mean values for each of the input variables over the observations within the clustering.

Consider, for example, a simple and small random subset of the *weather* dataset. This can be generated as below, where we choose only a small number of the available numeric variables:

```
> library(rattle)
> set.seed(42)
> obs1 <- sample(1:nrow(weather), 5)
> vars <- c("MinTemp", "MaxTemp",
            "Rainfall", "Evaporation")
> cluster1 <- weather[obs1, vars]
```

We now obtain the means of each of the variables. The vector of means then represents one of the clusters within our set of k clusters:

```
> mean(cluster1)
    MinTemp     MaxTemp    Rainfall Evaporation
       4.74       15.86        3.16        3.56
```

Another cluster will have a different mean:

```
> obs2 <- setdiff(sample(1:nrow(weather), 20), obs1)
> cluster2 <- weather[obs2, vars]
> mean(cluster2)
    MinTemp     MaxTemp    Rainfall Evaporation
     6.6474     19.7579      0.8421      4.4105
```

In comparing the two clusters, we might suggest that the second cluster generally has warmer days with less rainfall. However, without having actually built the clustering model, we can't really make too many such general observations without knowing the actual distribution of the observations.

A particular sentence in our knowledge representation language for k-means is then a collection of k sets of mean values for each of the variables. Thus, if we were to simply partition the *weather* dataset into ten sets (a common value for k), we would get ten sets of means for each of the four variables. Together, these 10 by 4 means represent a single sentence (or model) in the k-means "language."

9.2 Search Heuristic

For a given dataset, there are a very large number of possible k-means models that could be built. We might think to enumerate every possibility and then, using some measure that indicates how good the clustering is, choose the one that gets the best score. In general, this process of completely enumerating all possibilities would not be computationally possible. It may take hours, days, or weeks of computer time to generate and measure each possible set of clusters. Instead, the k-means algorithm uses a search heuristic. It begins with a random collection of k clusters. Each cluster is represented by a vector of the mean values for each of the variables.

The next step in the process is to then measure the distance between an observation and each of the k vectors of mean values. Each observation is then associated with its closest cluster.

We then recalculate the mean values based on the observations that are now associated with each cluster. This will provide us with a new collection of k vectors of means. With this new set of k means, we once again calculate the distance each observation is from each of the k means and reassociate the observation with the closest of the k means. This will often result in some observations moving from one group or cluster to another.

Once again, we recalculate the mean values based on the observations that are now associated with each cluster. Again, we have k new vectors of means. We repeat the process again. This iterative process is repeated until no more observations move from one cluster to another. The resulting clustering is then the model.

9.3 Measures

The basic measure used in building the model is a measure of distance, or conversely the measure of similarity between observations and the cluster means.

Any distance measure that measures the distance between two observations a and b must satisfy the following requirements:

$$d(a,b) \geq 0 \qquad \text{distance is nonnegative}$$

$$d(a,a) = 0 \qquad \text{distance to itself is 0}$$

$$d(a,b) = d(b,a) \qquad \text{distance is symmetric}$$

$$d(a,b) \leq d(a,c) + d(c,b) \qquad \text{triangular inequality}$$

One common distance measure is known as the Minkowski distance. This is formulated as

$$d(a,b) = \sqrt[q]{(|a_1 - b_1|^q + |a_2 - b_2|^q + \ldots + |a_n - b_n|^q)},$$

where a_1 is the value of variable 1 for observation a, etc. The value of q determines an actual distance formula. We can best picture the distance calculation using just two variables, like `MinTemp` and `MaxTemp`, from two observations. We plot the first two observations from the *weather* dataset in Figure 9.1 as generated using the following call to `plot()`. We also report the actual values being plotted.

```
> x <- round(weather$MinTemp[1:2])
> y <- round(weather$MaxTemp[1:2])
> plot(x, y, ylim=c(23, 29), pch=4, lwd=5,
       xlab="MinTemp", ylab="MaxTemp",
       bty="n")
> round(x)
[1]  8 14
> round(y)
[1] 24 27
```

When $q = 1$, d is known as the Manhattan distance:

$$d(a,b) = |a_1 - b_1| + |a_2 - b_2| + \ldots + |a_n - b_n|.$$

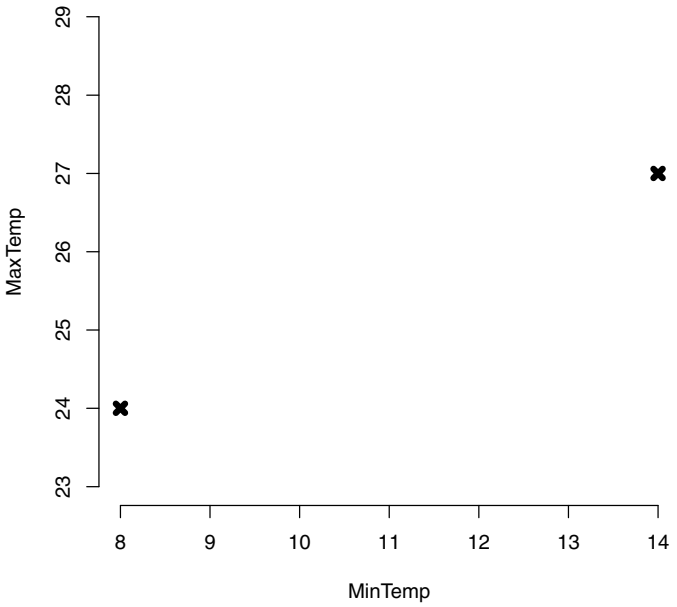


Figure 9.1: Two observations of two variables from the *weather* dataset. What are the possible ways of measuring the distance between these two points?

The Manhattan distance measure gets its name from one of the five boroughs of New York City. Most of the streets of Manhattan are laid out on a regular grid. Each block is essentially a rectangle. Figure 9.2 simplifies the grid structure but illustrates the point. Suppose we want to calculate the distance to walk from one block corner, say West 31st Street and 8th Avenue, to another, say West 17th Street and 6th Avenue. We must travel along the street, and the distance is given by how far we travel in each of just two directions, as is captured in the formula above.

For our *weather* dataset, we can add a `grid()` to the plot and limit our walk to the lines on the grid, as in Figure 9.2. The distance travelled will be $d = 6 + 3 = 9$, and one such path is shown as the horizontal and then vertical line in Figure 9.2.

When $q = 2$, d is known as the more familiar, and most commonly

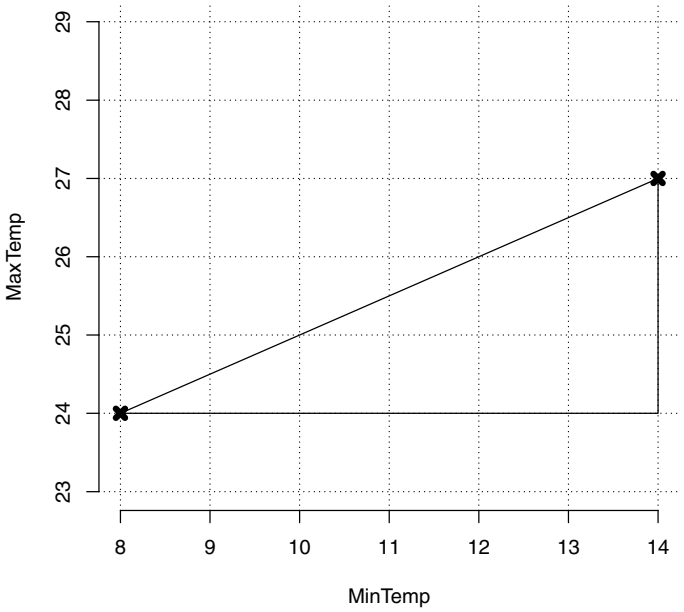


Figure 9.2: Measuring the distance by travelling the streets of Manhattan (the regular grid, with one path shown as the horizontal and then the vertical line), rather than as a bird might fly (the direct line between the two points).

used, Euclidean distance:

$$d(a,b) = \sqrt{(|a_1 - b_1|^2 + |a_2 - b_2|^2 + \ldots + |a_n - b_n|^2)}.$$

This is the straight-line distance between the two points shown in Figure 9.2. It is how a bird would fly direct from one point to another if it was flying high enough in Manhattan. The distance in this case is $d = \sqrt{6^2 + 3^2} = 6.32$.

In terms of how we measure the quality of the actual clustering model, there are very many possibilities. Most relate to measuring the distance between all of the observations within a cluster and summing that up. Then compare that with some measure of the distances between the means or even the observations of each of the different clusters. We will see and explain in a little more detail some of these measures in the next section.

9.4 Tutorial Example

The *weather* dataset is used to illustrate the building of a cluster model. The Cluster tab in the Rattle window provides access to various clustering algorithms, including k-means. `kmeans()` is provided directly through R by the standard **stats** package.

Building a Model Using Rattle

After loading a dataset into Rattle, we select the Cluster tab to be presented with various clustering algorithms. We will also see a simple collection of options available for use to fine-tune the model building. The k-means algorithm is the default option, and by default ten clusters will be built as the model. A random seed is provided. Changing the seed will result in a randomly different collection of starting points for our means. The heuristic search then begins the iterative process as described in Section 9.2.

Load the *weather* dataset from the Data tab, and then simply clicking the Execute button whilst on the Cluster tab will result in the k-means clustering output shown in Figure 9.3.

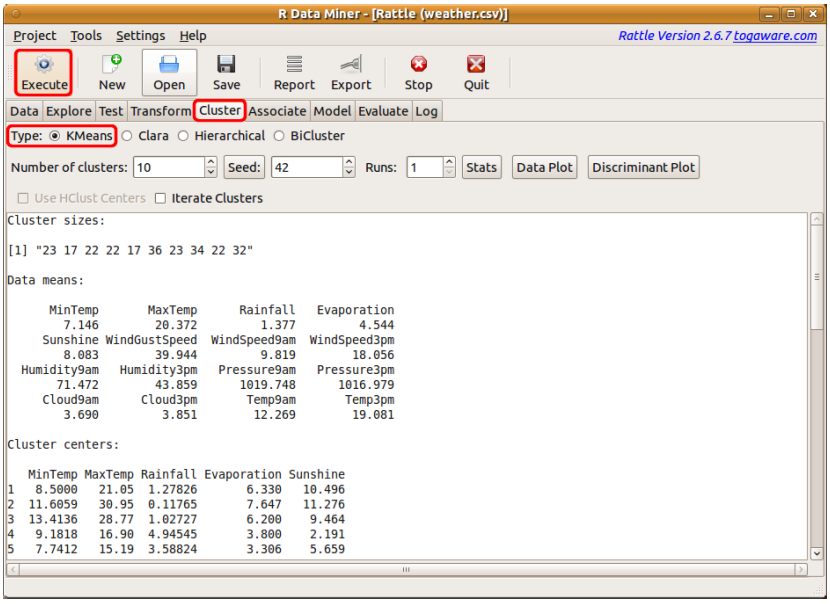


Figure 9.3: Building a k-means clustering model.

The text view contains a little information about the model that has been built. We will work our way through its contents. It begins with the cluster size, which is simply a count of the number of observations within each cluster:

```
Cluster sizes:

[1] "23 17 22 22 17 36 23 34 22 32"
```

Mean (or average) values are the basic representational language for models when using k-means. The text view provides a summary of the mean value of each variable over the whole dataset of observations (with the output truncated here):

```
Data means:

      MinTemp       MaxTemp      Rainfall   Evaporation
        7.146        20.372         1.377         4.544
     Sunshine WindGustSpeed  WindSpeed9am  WindSpeed3pm
        8.083        39.944         9.819        18.056
  Humidity9am   Humidity3pm   Pressure9am   Pressure3pm
       71.472        43.859      1019.748      1016.979
```

Cluster Means

A model from a k-means clustering point of view consists of ten (because ten clusters is the default) vectors of the mean values for each of the variables. The main content of the text view is a list of these means. We only show the first five variables and only eight of the ten clusters:

```
Cluster centers:

   MinTemp MaxTemp Rainfall Evaporation Sunshine
1   8.5000   21.05  1.27826       6.330   10.496
2  11.6059   30.95  0.11765       7.647   11.276
3  13.4136   28.77  1.02727       6.200    9.464
4   9.1818   16.90  4.94545       3.800    2.191
5   7.7412   15.19  3.58824       3.306    5.659
6   2.7667   17.16  0.66111       2.656    7.689
7  -0.7913   13.71  0.03478       1.922    7.496
8  11.3088   26.37  0.50000       6.288   10.259
9   1.5045   17.55  0.23636       3.500   10.223
10  7.8625   17.60  2.21875       4.519    6.122
```

Model Quality

The means are followed by a simple measure of the quality of the model:

```
Within cluster sum of squares:

 [1] 14460  5469  8062 11734 11062  9583  7258  7806  6146
[10] 11529
```

The measure used is the sum of the squares of the differences between the observations within each of the ten clusters.

Time Taken

Finally, we see how long the k-means algorithm took to build the ten clusters. For such a small dataset, very little time is required. The time taken is the amount of CPU time spent on the task:

```
Time taken: 0.00 secs
```

Tuning Options

The basic tuning option for building a k-means model in Rattle is simply the Number of clusters that are to be generated. The default is 10, but any positive integer greater than 1 is allowed.

Rattle also provides an option to iteratively build more clusters and measure the quality of each resulting model as a guide to how many clusters to build. This is chosen by enabling the Iterate Clusters option. When active, a model with two clusters, then a model with three clusters, and so on up to a model with ten (or as many as specified) clusters will be built. A plot is generated and displayed to report the improvement in the quality measure (the sum of the within cluster sum of squares).

As mentioned previously, the Seed option allows different starting points to be used for the heuristic search. Each time a different seed is used, the resulting model will usually be different.

For some datasets, differences between the models using different seeds will often not be too large, though for others they might be quite large. In the latter case, we are finding different, possibly less optimal or perhaps equally optimal models each time. The Runs option will repeat the model building the specified number of times and choose the model

that provides the best performance against the measure of model quality. For each different seed, we can check the list of cluster size to confirm that we obtain a collection of clusters that are about the same sizes each time, though the order in the listing changes.

Once a model has been built, the Stats, Data Plot, and Discriminant Plot buttons become available. Clicking the Stats button will result in quite a few additional cluster statistics being displayed in the text view. These can all participate in determining the quality of the model and comparing one k-means model against another. The Data Plot and the Discriminant Plot buttons result in plots that display how the clusters are distributed across the data. The discriminant coordinates plot is generated by projecting the original data to display the key differences between clusters, similar to principal components analysis. The plots are probably only useful for smaller datasets (in the hundreds or thousands).

The Rattle user interface also provides access to the Clara, Hierarchical, and BiCluster clustering algorithms. These are not covered here.

Building a Model Using R

The primary function used within R for k-means clustering is `kmeans()` which comes standard with R. We can build a k-means cluster model using the encapsulation idea presented in Section 2.9:

```
> weatherDS <- new.env()
```

From the *weather* dataset, we will select only two numeric variables on which to cluster, and we also ignore the output variable `RISK_MM`:

```
> library(rattle)
> evalq({
    data <- weather
    nobs <- nrow(data)
  }, weatherDS)
```

We now create a model container to store the results of the modelling and build the actual model. The container also includes the weatherDS dataset information.

```
> weatherKMEANS <- new.env(parent=weatherDS)
> evalq({
    model <- kmeans(x=na.omit(data[, vars]), centers=10)
  }, weatherKMEANS)
```

We have used `kmeans()` and passed to it a dataset with any observations having missing values omitted. The function otherwise complains if the data contains missing values, as we might expect when using a distance measure. The `centers=` option is used either to specify the number of clusters or to list the starting points for the clustering.

9.5 Discussion

Number of Clusters

The primary tuning parameter for the k-means algorithm is the number of clusters, k. Simply because the default is to identify ten clusters does not mean that 10 is a good choice at all. Choosing the number of clusters is often quite a tricky exercise. Sometimes it is a matter of experimentation and other times we might have some other knowledge to help us decide. We will soon note that the larger the number of clusters relative to the size of the sample, the smaller our clusters will generally be. However, a common observation is that often we might end up with a small number of clusters containing most of the observations and a large number of clusters containing only a few observations each.

We also note that different cluster algorithms (and even simply using different random seeds to initiate the clustering) can result in different (and sometimes very different) clusters. How much they differ is a measure of the stability of the clustering.

Rattle provides an Iterate Clusters option to assist with identifying a good number of clusters. The approach is to iterate through different values of k. For each k, we observe the sum of the within cluster sum of squares. A plot is generated to show both the sum of squares and its change in the sum of squares. A heuristic is to choose the number of clusters where we see the largest drop in the sum of the within cluster sum of squares.

Shape of Clusters

One of the characteristics to distinguish between clustering algorithms is the shape of the resulting clusters. Essentially, the k-means algorithm, as with any algorithm that uses the distance to a mean as the representation of the clusters, produces convex clusters. Other clustering algorithms ex-

ist that can produce differently shaped clusters that might better reflect the data.

Other Cluster Algorithms

R supports a very large variety of clustering algorithms besides the k-means algorithm we have described here. They are grouped into the partitioning type of algorithms, of which k-means is one example, model-based algorithms (see **mclust** (Fraley and Raftery, 2006), for example), and hierarchical clustering (see `hclust()` from **stats** and `agnes()` for agglomerative clustering, and `diana()` for divisive clustering from **cluster** (Maechler et al., 2005)). Rattle supports the building of hierarchical clusters using `hclust()`. Such an algorithm builds the clusters iteratively and hierarchically.

For an agglomerative hierarchical approach, the two closest observations form the first cluster. Then the next two closest observations, but now also including the mean of the first cluster as a "combined" observation, form the second cluster, and so on until we have formed a single cluster. The resulting collection of potential clusters can be drawn using a dendrogram, as shown in Figure 9.4.

An advantage of this approach is that we get a visual clue as to the number of clusters that naturally appear in the data. In Figure 9.4 we have drawn boxes to indicate perhaps three clusters. A disadvantage is that this approach is really only useful for a small dataset.

Recent research has explored the issue of very high dimensional data, or data with very many variables. For such data the k-means algorithm performs rather poorly, as all observations essentially become equidistant from each other. A successful approach has been developed (Jing et al., 2007) using a weighted distance measure. The algorithm essentially chooses only subsets of the variables on which to cluster. This has been referred to as subspace clustering.

The **siatlust** (Williams et al., 2011) package, provides an implementation of this modification to the k-means algorithm. Entropy weighted variable selection through `ewkm()` is used to improve the clustering performance with high dimensional data.

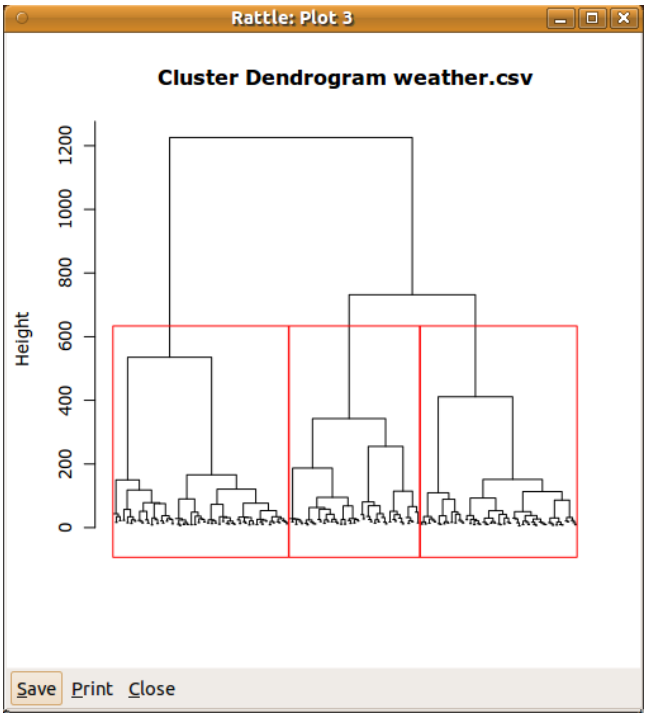


Figure 9.4: A sample dendrogram showing three clusters.

9.6 Command Summary

This chapter has referenced the following R packages, commands, functions, and datasets:

`agnes()`	function	An agglomerative clustering algorithm.
cluster	package	A variety of tools for cluster analysis.
`diana()`	function	A divisive clustering algorithm.
`ewkm()`	function	Entropy weighted k-means.
`evalq()`	function	Access environment for storing data.
`grid()`	command	Add a grid to a plot.
`hclust()`	function	A hierarchical clustering algorithm.
`kmeans()`	function	The k-means clustering algorithm.
`mean`	function	Calculate the mean values.

`plot()`	command	Draw a dendrogram for an hclust object.
`round()`	function	Round numbers to specific digits.
`set.seed()`	command	Reset random sequence for sampling.
siatclust	package	Weighted and subspace k-means.
stats	package	Base package providing k-means.
weather	dataset	Sample dataset from **rattle**.