Basic Clustering Example

D2K Course Staff

February 13 2019

Introduction

The general idea behind cluster analysis is to group observations together such that members of each group are more similar to each other than members in different groups. In this way, we might be able to find some underlying group structure in the observations based on the collected variables. This can then be used to infer possible similar behaviors or functions amongst all observations in a cluster; or example, we might be interested in finding functional groups of proteins in genetics data, or species of plants that behave similarly under different habitat conditions.

Below, we introduce two of the most commonly used and basic clustering methods, k means and hierarchical clustering. In this example, we will analyze the `wine` data set from the `rattle.data` package. We will be clustering assuming that we do not know the wine type, and we will attempt to see if we can find different groups of wines in the data, perhaps to see if we can find classifications to ensure similar taste or composition amongst all wines of the same labeled type.

```r
library(rattle.data)
data(wine)
head(wine)
```

```
  Type Alcohol Malic  Ash Alcalinity Magnesium Phenols Flavanoids
1    1   14.23  1.71 2.43       15.6       127    2.80       3.06
2    1   13.20  1.78 2.14       11.2       100    2.65       2.76
3    1   13.16  2.36 2.67       18.6       101    2.80       3.24
4    1   14.37  1.95 2.50       16.8       113    3.85       3.49
5    1   13.24  2.59 2.87       21.0       118    2.80       2.69
6    1   14.20  1.76 2.45       15.2       112    3.27       3.39
  Nonflavanoids Proanthocyanins Color  Hue Dilution Proline
1          0.28            2.29  5.64 1.04     3.92    1065
2          0.26            1.28  4.38 1.05     3.40    1050
3          0.30            2.81  5.68 1.03     3.17    1185
4          0.24            2.18  7.80 0.86     3.45    1480
5          0.39            1.82  4.32 1.04     2.93     735
6          0.34            1.97  6.75 1.05     2.85    1450
```

K Means

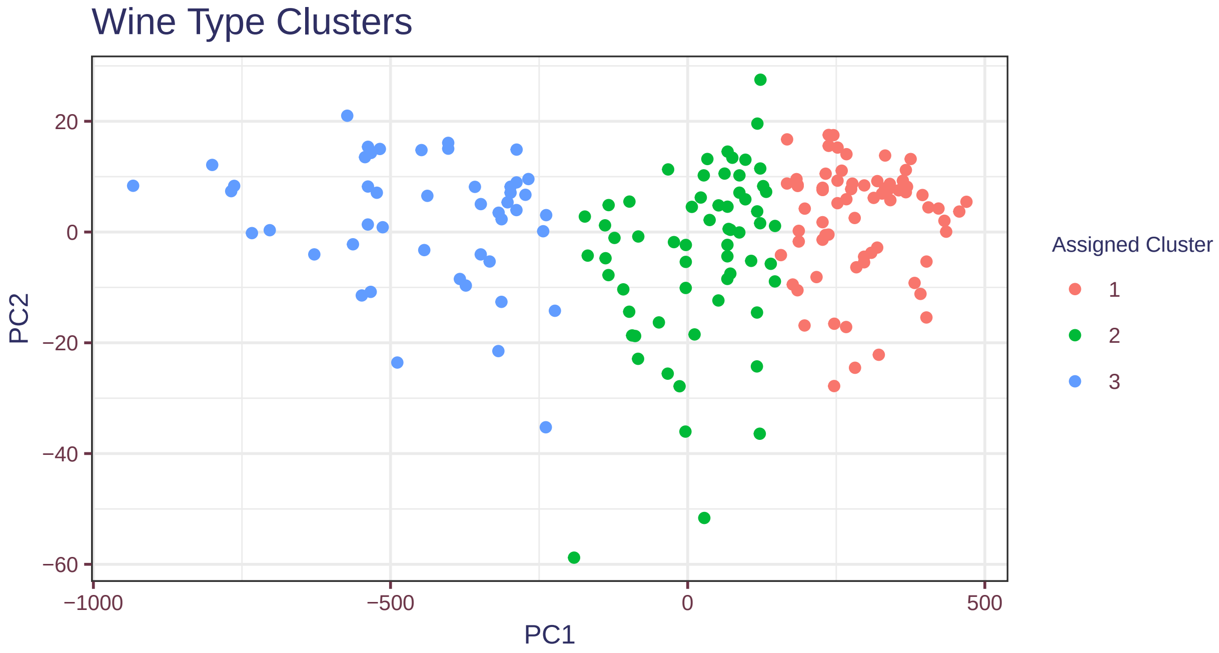
For k means clustering, the objective is to minimize the dostances betweem the points and their assigned cluster centroids. the user must choose the number of clusters desired. The algorithm proceeds as follows:

1. Randomly assign each observation to a cluster.
2. Calculate the centroid for each cluster.
3. Assign each point to the closest centroid.
4. Repeat steps 2 and 3 until convergence (i.e., there are no changes in the assignments of points to clusters.)

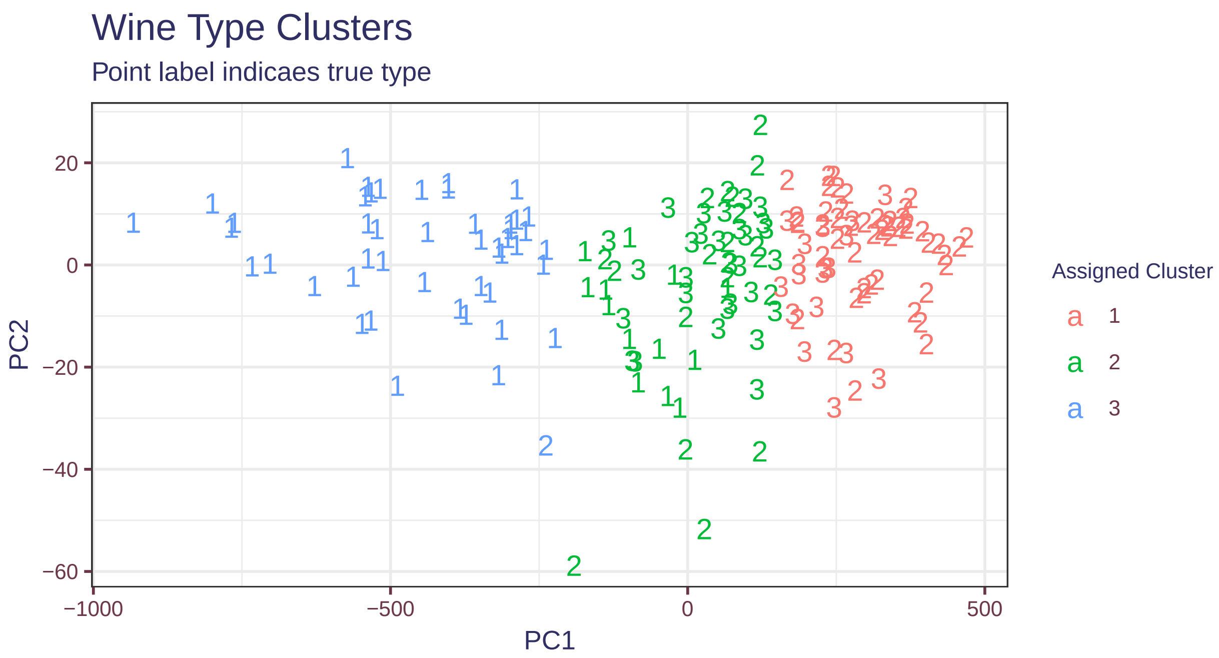
The result of running k-means on the `wine` data set with $k = 3$ are shown below. This can be done with the `kmeans` function in base R. (This is cheating a little because we know there are 3 true types in the data.)

```
kmeans_1 <- kmeans(wine[, -1], 3, iter.max = 100)
```

(Note here that the data are plotted by the first 2 principal components, which tends to make the separation more clear visually. The clusters would almost certainly not be as easily seen if we plotted by two raw variables.)



SInce we actually have the underlying truth, we can see if these cluster assignments are good. Let’s see then!



There does seem to be a 3 clusters that are mainly composed of one of the 3 different types. However, there is some mixing.

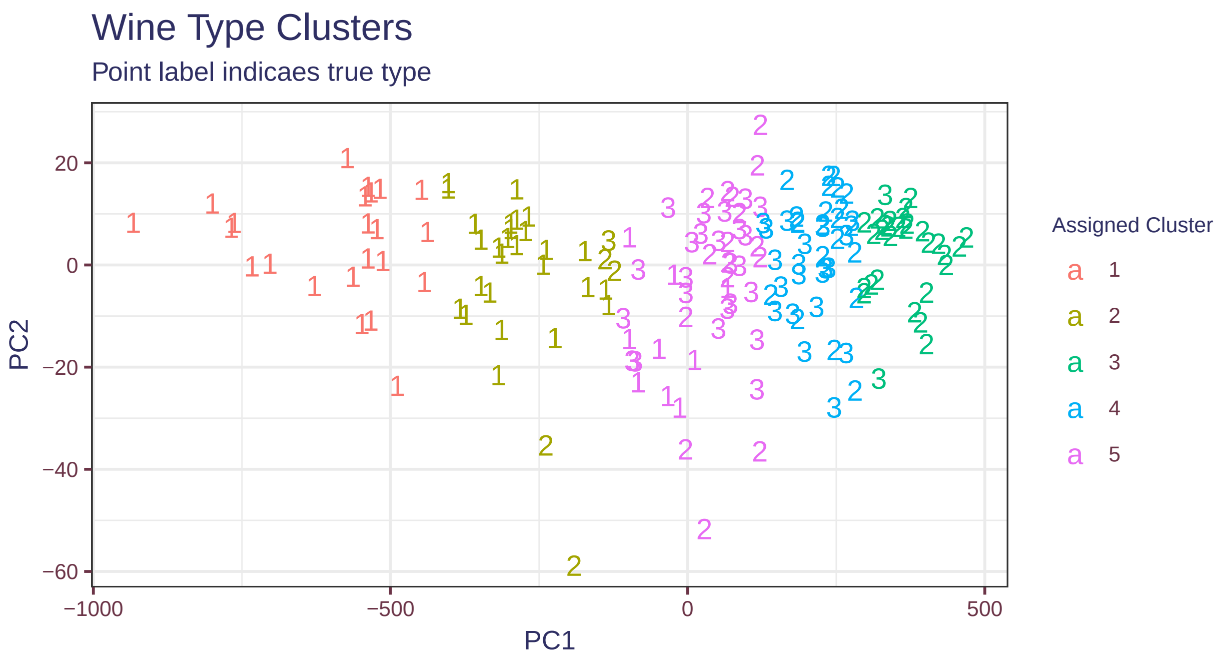
Disadvantages

K means clustering has (at least) 3 main disadvantages:

1. The results depend on the starting random assignment of points in the initialization of the algorithm. Because the algoirthm is greedy, it does not check for an overall global optimum.
2. There is really no way to choose the number of clusters in a data driven manner. Since the ultimate goal of k-means clustering is to reduce the distance between clusters and their centroids, more clusters will always appear objectively better to the algorithm. If you choose the wrong number of clusters, you're out of luck.
3. Creating more clusters does not respect the structure of the cluster analysis for smaller k, e.g. a four cluster result does not simply split one of the clusters from a 3 cluster result. This means that two points in different clusters for $k = 3$ may be in the same cluster when $k = 4$ - probably not what we want!

Let's see what happens if we run the above example for $k = 5$ clusters:

```
kmeans_2 <- kmeans(wine[, -1], 5, iter.max = 100)
```



If you compare to the above, points that were assigned to different clusters when $k = 3$ are now in the same cluster when $k = 5$. We also do not appear to get improved performance in terms of correctly grouping the observations in to their true types.

Hierarchical Clustering

Hierarchical clustering, on the other hand, grows clusters from individual data points sequentially by linking points or clusters together via the distances between them. One advantage of hierarchical clustering is the hierarchical structure - removing one cluster in your final result simply combines the two closest clusters in to one, which seems much more intuitive. Another plus is that we can see the distances at which points in clusters are linked.

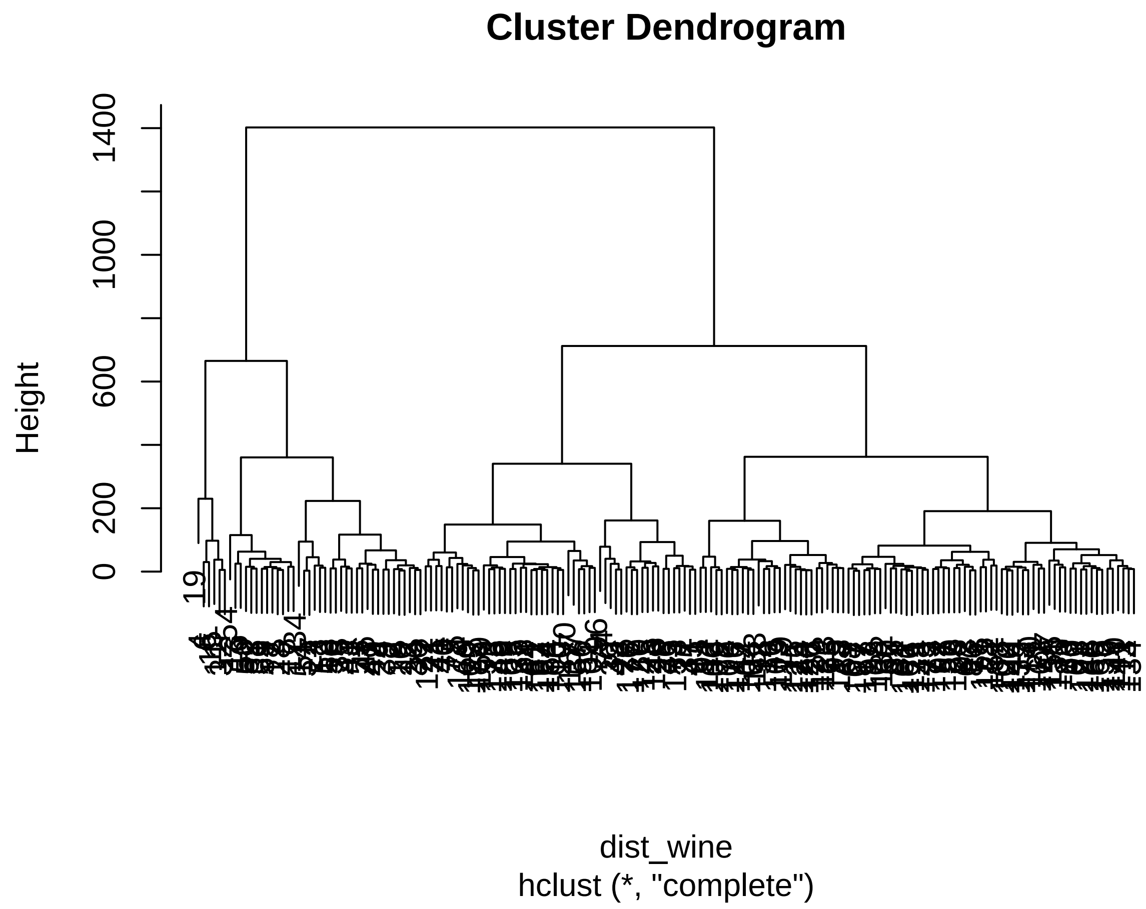
The result of running hierarchical clustering on the `wine` data set are shown below. Note the steps of the algorithm:

1. Calculate a distance matrix using `dist`in base R.
2. Run the hierarchical clustering using `hclust` in base R.

The results are shown in a dendogram. The individual observations are on the x-axis, and the distance is on the y-axis. The tree structure in the plot shows the distance at which two different clusters are linked in to one - the higher up the tree, the further away the linkage distance is.

```
dist_wine <- dist(wine[, -1])

hclust_1 <- hclust(dist_wine)
plot(hclust_1)
```



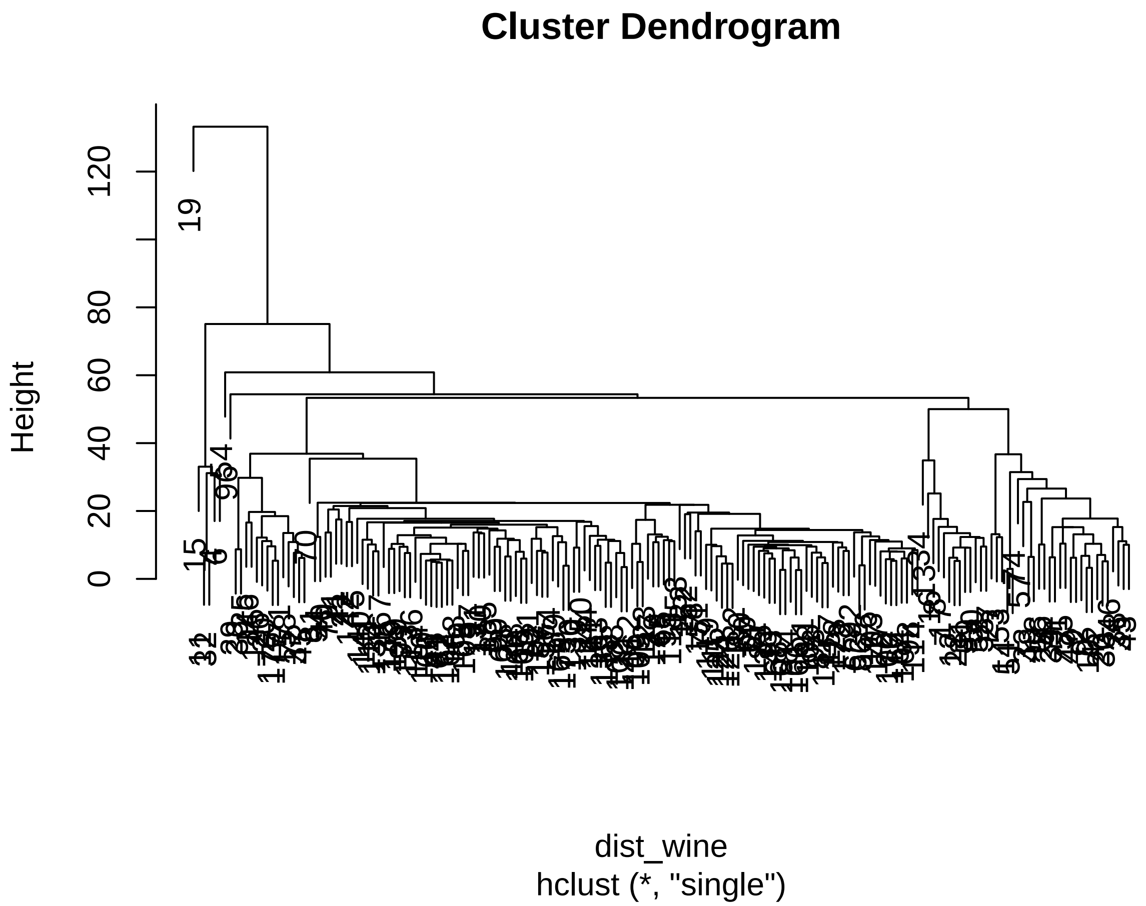
We then choose the number of clusters by either choosing the number of desired clusters or choosing a cutoff point for the maximim distance we allow for two points to be in the same cluster. For example, at a chosen cutoff of 600, we will have 4 clusters. (The actual numeric value of the distance doesn't really have meaning itself, so don't worry too much about it.)

Disadvantages

1. The choice of the number of clusters is still ultimately an arbitrary decision.
2. Your cluster results depend on your choice of linkage distance. There are a few options here: complete linkage (the default option) will join points and clusters based on the maximum distance between points contained in each, single linkage will join points and clusters based on the minimum distance between points in each, and average linkage will join points and clusters based on the average distance between all the points in each. This choice can vastly affect the outcome of clustering.
3. Calculating distances is computationally intensive, especially in high dimensions. This is where using dimension reduction for preprocesing can help greatly.

Let's see what happens if we run the above example with single linkage:

```r
hclust_2 <- hclust(dist_wine, method = "single")
plot(hclust_2)
```



The same choices of number of clusters or cutoffs with create vastly different clusters compared to complete linkage!