Comparitive Study of Cluster Analysis Approaches

Arjun Dutta

3 March 2019

1. Segmentation Philosophy

The general goal of market segmentation is to find groups of customers that differ in important ways associated with product interest, market participation, or response to marketing efforts.

2. What is Segmentation/Clustering?

Segmentation is like slicing a pie, and any pie might be sliced in an infinite number of ways.

We explore two distinct yet related areas of statistics: Clustering or cluster analysis and Classification. These are the primary branches of what is sometimes called statistical learning.

What is Supervised Learning ?

In supervised learning, a model is presented with observations whose outcome status (dependent variable) is known, with a goal to predict that outcome from the independent variables.

What is Un-Supervised Learning ?

Un-Supervised learning we do not know the outcome groupings but attempt to discover them from structure in the data.

Load and Explore the Data

```
seg.raw <- read.csv("http://goo.gl/qw303p")
seg.df <- seg.raw[ , -7] # remove the known segment assignments
summary(seg.df)
```

```
##       age           gender        income            kids        ownHome
##  Min.   :19.26   Female:157   Min.   : -5183   Min.   :0.00   ownNo :159
##  1st Qu.:33.01   Male  :143   1st Qu.: 39656   1st Qu.:0.00   ownYes:141
##  Median :39.49                Median : 52014   Median :1.00
##  Mean   :41.20                Mean   : 50937   Mean   :1.27
##  3rd Qu.:47.90                3rd Qu.: 61403   3rd Qu.:2.00
##  Max.   :80.49                Max.   :114278   Max.   :7.00
##   subscribe
##  subNo :260
##  subYes: 40
##
##
##
##
```

Steps of Clustering

Clustering analysis requires two stages: finding a proposed cluster solution and evaluating that solution for one's business needs.

For each method we go through the following steps:

1. Transform the data if needed for a particular clustering method; for instance, some methods require all numeric data (e.g., kmeans(), mclust()) or all categorical data (e.g., poLCA()).
2. Compute a distance matrix if needed; some methods require a precomputed matrix of similarity in order to group observations (e.g., hclust()) .
3. Apply the clustering method and save its result to an object. For some methods this requires specifying the number (K) of groups desired (e.g., kmeans(), poLCA()).
4. For some methods, further parse the object to obtain a solution with K groups (e.g., hclust()).
5. Examine the solution in the model object with regard to the underlying data, and consider whether it answers a business question.

```
seg.summ <- function(data, groups) {aggregate(data, list(groups), function(x) mean(as.numeric(x)))}
```

Heirarchial Clustering

Hierarchical clustering is a popular method that groups observations according to their similarity.

hclust() is a distance-based algorithm that operates on a dissimilarity matrix, an N-by-N matrix that reports a metric for the distance between each pair of observations.

How it works

The hierarchical clustering method begins with each observation in its own cluster. It then successively joins neighboring observations or clusters one at a time according to their distances from one another, and continues this until all observations are linked. This process of repeatedly joining observations and groups is known as an agglomerative method.

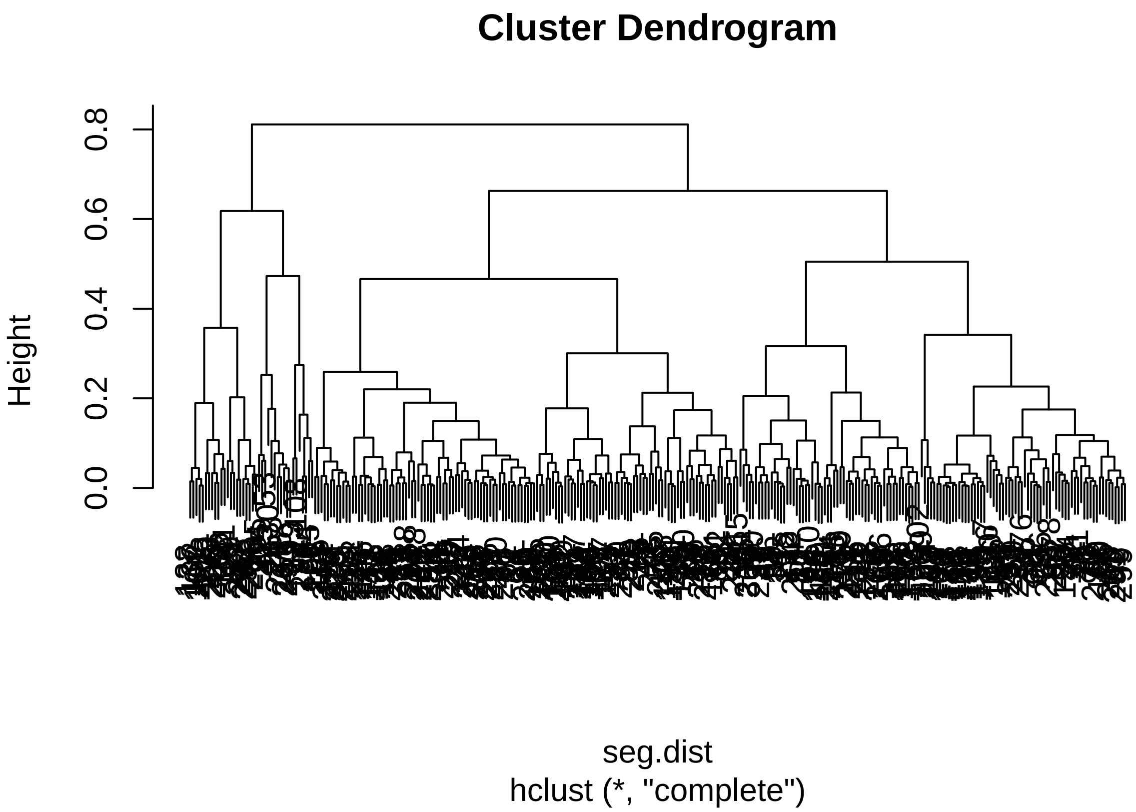
There are many ways to compute distance, and we start by examining the best-known method, the Euclidean distance.

The daisy() function in the cluster package [108] works with mixed data types by rescaling the values, so we use that instead of Euclidean distance:

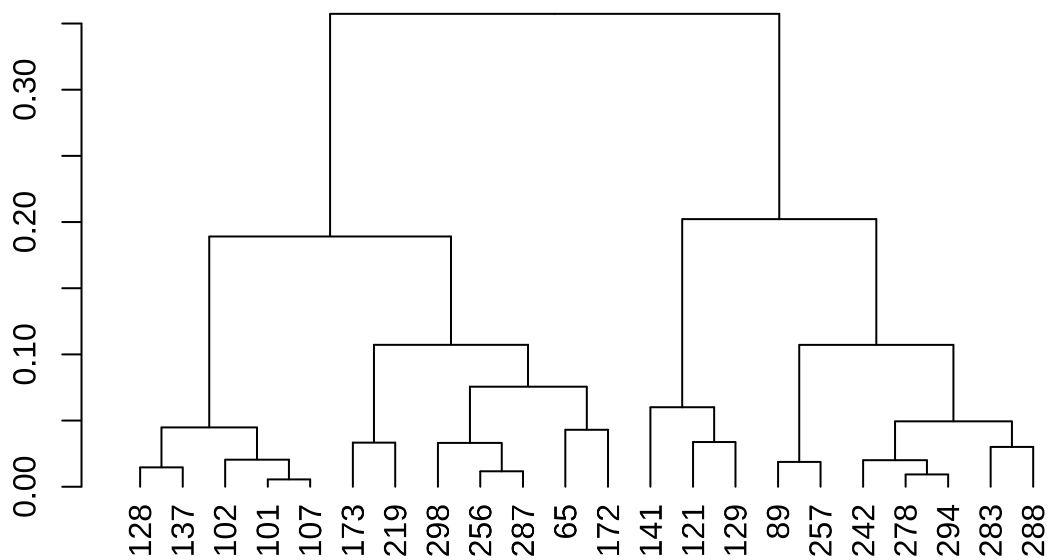
```
library(cluster) # daisy works with mixed data types
seg.dist <- daisy(seg.df)
seg.hc <- hclust(seg.dist, method="complete")
```

We use the complete linkage method, which evaluates the distance between every member when combining observations and groups.

```
plot(seg.hc)
```




```
plot(cut(as.dendrogram(seg.hc), h=0.5)$lower[[1]])
```



Finally, we might check one of the goodness-of-fit metrics for a hierarchical cluster solution. One method is the cophenetic correlation coefficient (CPCC), which assesses how well a dendrogram (in this case seg.hc) matches the true distance metric (seg.dist) [145]. We use cophenetic() to get the distances from the dendrogram, and compare it to the dist() metrics with cor():

```
cor(cophenetic(seg.hc), seg.dist)
```

```
## [1] 0.7682436
```

CPCC is interpreted similarly to Pearson's r. In this case, CPCC > 0.7 indicates a relatively strong fit, meaning that the hierarchical tree represents the distances between customers well.

Kmeans

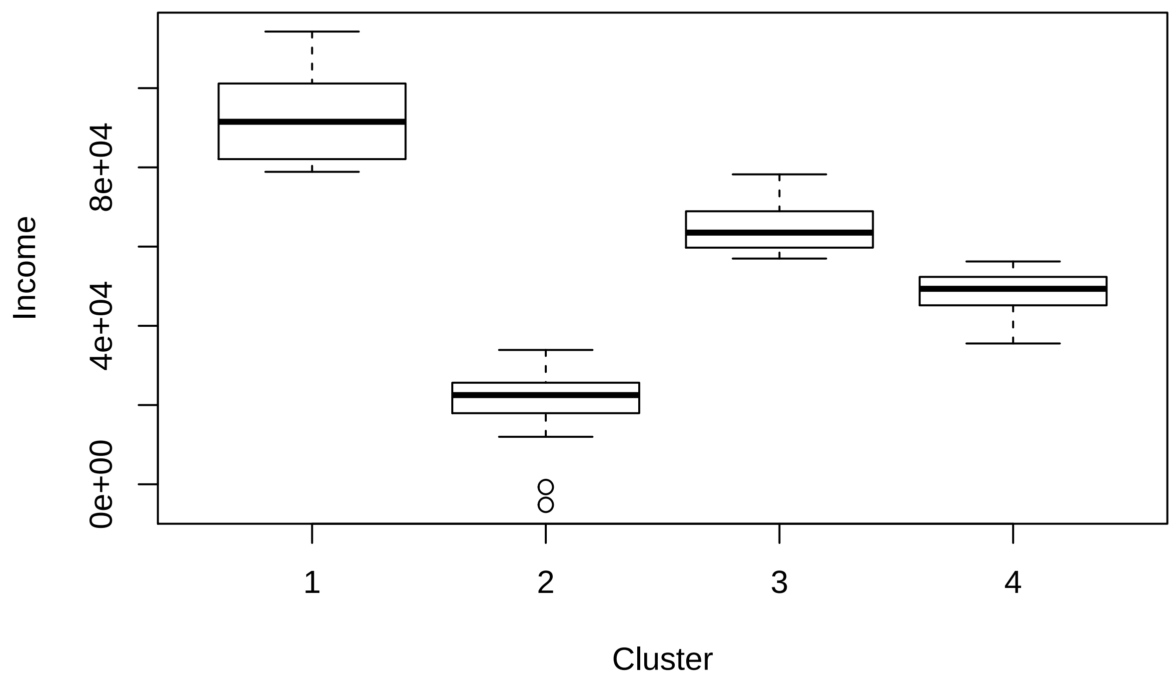
K-means clustering attempts to find groups that are most compact, in terms of the mean sum-of-squares deviation of each observation from the multivariate center (centroid) of its assigned group.

```
seg.df.num <- seg.df
seg.df.num$gender <- ifelse(seg.df$gender=="Male", 0, 1)
seg.df.num$ownHome <- ifelse(seg.df$ownHome=="ownNo", 0, 1)
seg.df.num$subscribe <- ifelse(seg.df$subscribe=="subNo", 0, 1)
summary(seg.df.num)
```

```
##       age            gender           income            kids
##  Min.   :19.26   Min.   :0.0000   Min.   : -5183   Min.   :0.00
##  1st Qu.:33.01   1st Qu.:0.0000   1st Qu.: 39656   1st Qu.:0.00
##  Median :39.49   Median :1.0000   Median : 52014   Median :1.00
##  Mean   :41.20   Mean   :0.5233   Mean   : 50937   Mean   :1.27
```

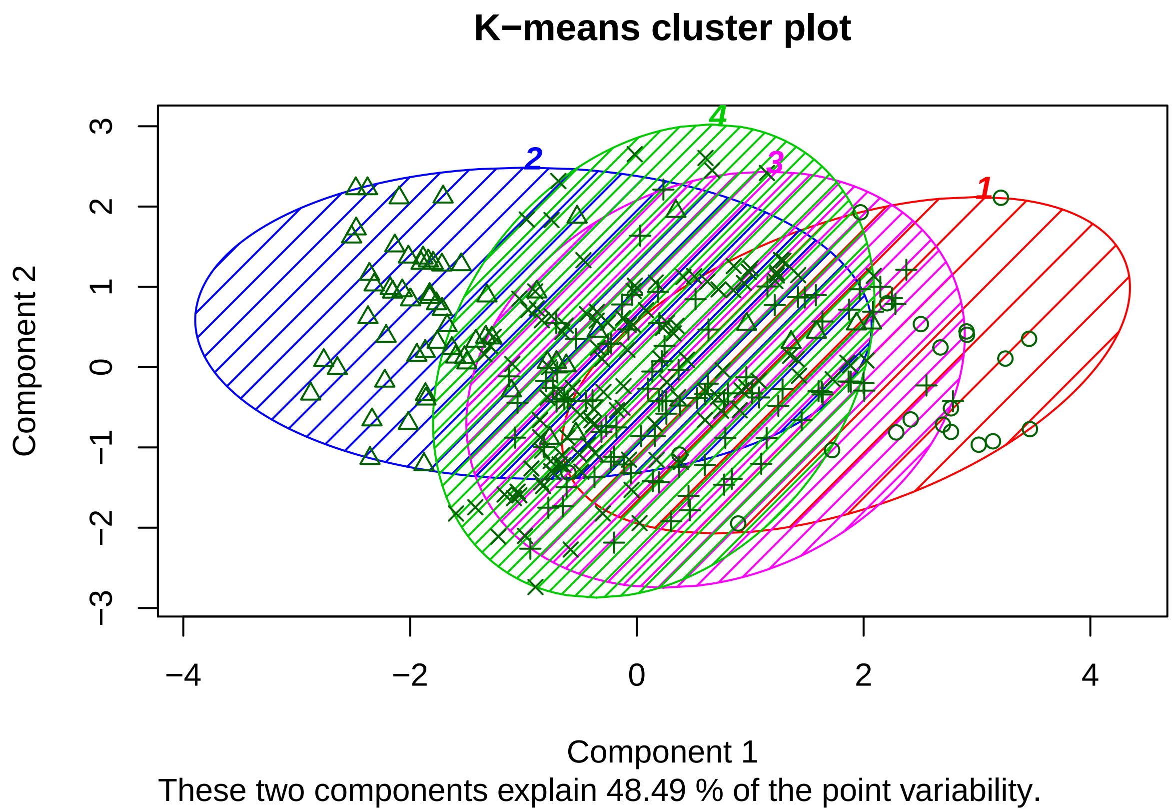
```
##  3rd Qu.:47.90   3rd Qu.:1.0000   3rd Qu.: 61403   3rd Qu.:2.00
##  Max.   :80.49   Max.   :1.0000   Max.   :114278   Max.   :7.00
##     ownHome        subscribe
##  Min.   :0.00   Min.   :0.0000
##  1st Qu.:0.00   1st Qu.:0.0000
##  Median :0.00   Median :0.0000
##  Mean   :0.47   Mean   :0.1333
##  3rd Qu.:1.00   3rd Qu.:0.0000
##  Max.   :1.00   Max.   :1.0000
```

```r
set.seed(96743)
seg.k <- kmeans(seg.df.num, centers=4)
boxplot(seg.df.num$income ~ seg.k$cluster, ylab="Income", xlab="Cluster")
```




```r
library(cluster)

clusplot(seg.df, seg.k$cluster, color=TRUE, shade=TRUE,labels=4, lines=0, main="K-means cluster plot")
```



Model-Based Clustering: Mclust()

The key idea for model-based clustering is that observations come from groups with different statistical distributions (such as different means and variances).

Such models are also known as "mixture models" because it is assumed that the data reflect a mixture of observations drawn from different populations, although we don't know which population each observation was drawn from.

As you might guess, because mclust models data with normal distributions, it uses only numeric data.

```
library(mclust)
```

```
## Package 'mclust' version 5.4.2
## Type 'citation("mclust")' for citing this R package in publications.
```

```
seg.mc <- Mclust(seg.df.num)
summary(seg.mc)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 3 components:
##
##  log.likelihood   n df       BIC       ICL
##       -5137.106 300 73 -10690.59 -10690.59
##
```

```
## Clustering table:
##   1   2   3
## 163  71  66
```

This tells us that the data are estimated to have three clusters (components) with the sizes as shown in the table. Mclust() compared a variety of different mixture shapes and concluded that an ellipsoidal model (modeling the data as multivariate ellipses) fit best.

Now, We try a 4-cluster solution by telling Mclust() the number of clusters we want with the G=4 argument:

```
seg.mc4 <- Mclust(seg.df.num, G=4)
summary(seg.mc4)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VII (spherical, varying volume) model with 4 components:
##
##  log.likelihood   n df       BIC       ICL
##       -16862.69 300 31 -33902.19 -33906.18
##
## Clustering table:
##   1   2   3   4
## 104  66  59  71
```

We compare the 3-cluster and 4-cluster models using the Bayesian information criterion (BIC) [129] with BIC(model1, model2):

```
BIC(seg.mc, seg.mc4)
```

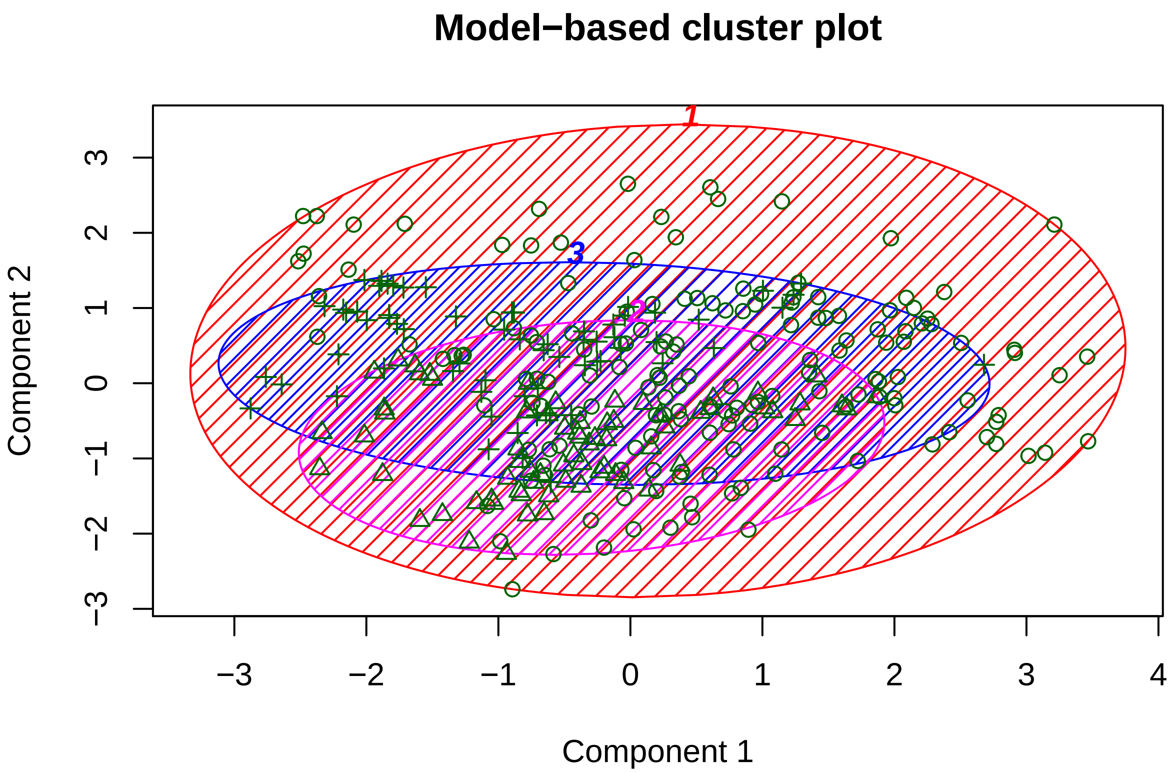
```
##         df      BIC
## seg.mc  73 10690.59
## seg.mc4 31 33902.19
```

The difference between the models is 181. The key point to interpreting BIC is to remember this: the lower the value of BIC, on an infinite number line, the better. BIC of ???1,000 is better than BIC of ???990; and BIC of 60 is better than BIC of 90. There is one important note when interpreting BIC in R: unfortunately, some functions return the negative of BIC, which would then have to be interpreted in the opposite direction.We see above that BIC() reports positive values while Mclust() returns the same values in the negative direction. If you are ever unsure of the direction to interpret, use the BIC() function and interpret as noted (lower values are better). Alternatively, you could also check the log-likelihood values, where higher log-likelihood values are better (e.g., ???1,000 is better than ???1,100).

```
seg.summ(seg.df, seg.mc$class)
```

```
##   Group.1      age   gender   income     kids  ownHome subscribe
## 1       1 44.68018 1.472393 52980.52 1.171779 1.865031  1.245399
## 2       2 38.02229 1.000000 51550.98 1.422535 1.000000  1.000000
## 3       3 36.02187 2.000000 45227.51 1.348485 1.000000  1.000000
```

```
clusplot(seg.df, seg.mc$class, color=TRUE, shade=TRUE, labels=4, lines=0, main="Model-based cluster plot
```



These two components explain 48.49 % of the point variability.