Partitioning & Hierarchical Clustering

Likhita Pula

6/4/2021

Importing data for analysis For the first assignment on Partitioning and hierarchical clustering I have choosen a dataset that describes various attributes of wine. This dataset is hosted on www.kaggle.com.

```
wine_data <- read.delim("C:\\Users\\likhi\\Desktop\\MSDA\\Data Mining 2\\Dataset\\wine-clustering.csv",
                        header = TRUE,
                        sep=",")
```

Exploratory data analysis As a next step, we can look at the overview and statistical details of the dataset used.

Below we can see the top 6 rows of the data set

```
head(wine_data)
```

```
##   Alcohol Malic_Acid  Ash Ash_Alcanity Magnesium Total_Phenols Flavanoids
## 1   14.23       1.71 2.43         15.6       127          2.80       3.06
## 2   13.20       1.78 2.14         11.2       100          2.65       2.76
## 3   13.16       2.36 2.67         18.6       101          2.80       3.24
## 4   14.37       1.95 2.50         16.8       113          3.85       3.49
## 5   13.24       2.59 2.87         21.0       118          2.80       2.69
## 6   14.20       1.76 2.45         15.2       112          3.27       3.39
##   Nonflavanoid_Phenols Proanthocyanins Color_Intensity  Hue OD280 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
```

Checking if the data set has any missing values:

```
sum(is.na(wine_data))
```

```
## [1] 0
```

We can see that there are no null values in the data.

The wine dataset had below features:

- Alcohol

- Malic acid
- Ash
- Alcalinity of ash
- Magnesium
- Total phenols
- Flavanoids
- Nonflavanoid phenols
- Proanthocyanins
- Color intensity
- Hue
- OD280 of diluted wines
- Proline

Checking the dimension of the dataset:

```
dim(wine_data)
```

```
## [1] 178  13
```

we can see that there are a total of 178 rows and 13 columns (features) in the wine dataset.

Checking the statistical attributes like maximum, minimum, mean etc. of the dataset features.

```
summary(wine_data)
```

```
##     Alcohol        Malic_Acid         Ash         Ash_Alcanity  
##  Min.   :11.03   Min.   :0.740   Min.   :1.360   Min.   :10.60  
##  1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210   1st Qu.:17.20  
##  Median :13.05   Median :1.865   Median :2.360   Median :19.50  
##  Mean   :13.00   Mean   :2.336   Mean   :2.367   Mean   :19.49  
##  3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558   3rd Qu.:21.50  
##  Max.   :14.83   Max.   :5.800   Max.   :3.230   Max.   :30.00  
##    Magnesium      Total_Phenols     Flavanoids    Nonflavanoid_Phenols
##  Min.   : 70.00   Min.   :0.980   Min.   :0.340   Min.   :0.1300      
##  1st Qu.: 88.00   1st Qu.:1.742   1st Qu.:1.205   1st Qu.:0.2700      
##  Median : 98.00   Median :2.355   Median :2.135   Median :0.3400      
##  Mean   : 99.74   Mean   :2.295   Mean   :2.029   Mean   :0.3619      
##  3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.875   3rd Qu.:0.4375      
##  Max.   :162.00   Max.   :3.880   Max.   :5.080   Max.   :0.6600      
##  Proanthocyanins Color_Intensity       Hue             OD280      
##  Min.   :0.410   Min.   : 1.280   Min.   :0.4800   Min.   :1.270  
##  1st Qu.:1.250   1st Qu.: 3.220   1st Qu.:0.7825   1st Qu.:1.938  
##  Median :1.555   Median : 4.690   Median :0.9650   Median :2.780  
##  Mean   :1.591   Mean   : 5.058   Mean   :0.9574   Mean   :2.612  
##  3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.1200   3rd Qu.:3.170  
##  Max.   :3.580   Max.   :13.000   Max.   :1.7100   Max.   :4.000  
##     Proline      
##  Min.   : 278.0  
##  1st Qu.: 500.5  
##  Median : 673.5  
##  Mean   : 746.9  
##  3rd Qu.: 985.0  
##  Max.   :1680.0  
```

We can see that few attributes like “proline” or “Magnesium” have larger values compared to other attributes

These attributes / features with larger variance can substantially influence output clusters, thus it would be good to scale our dataset.

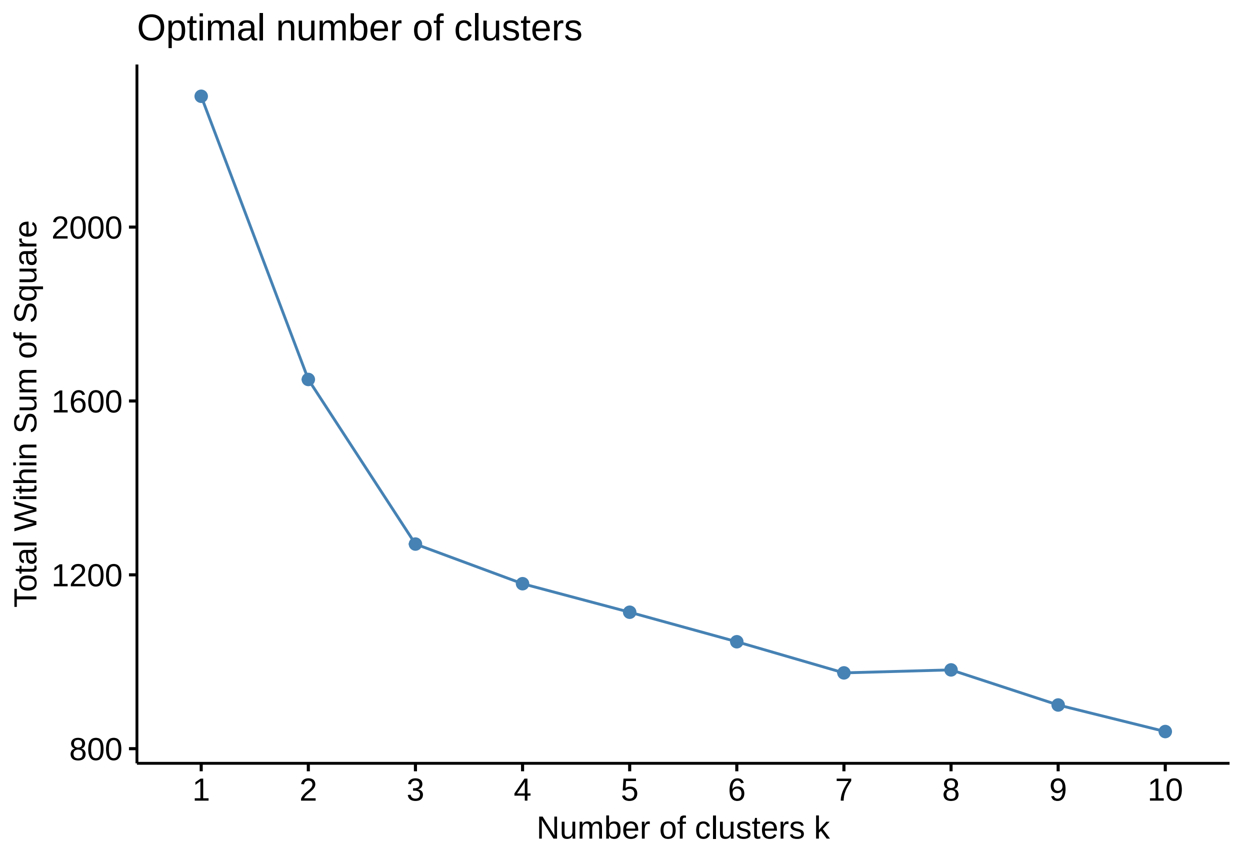
```r
scale_M <- scale(
  x = wine_data
)
```

scaling the data for analysis

Partitioning Clustering

Finding the optimal number of clusters We can use the Elbow method (sum of squares) method to find the optimal number of clusters:

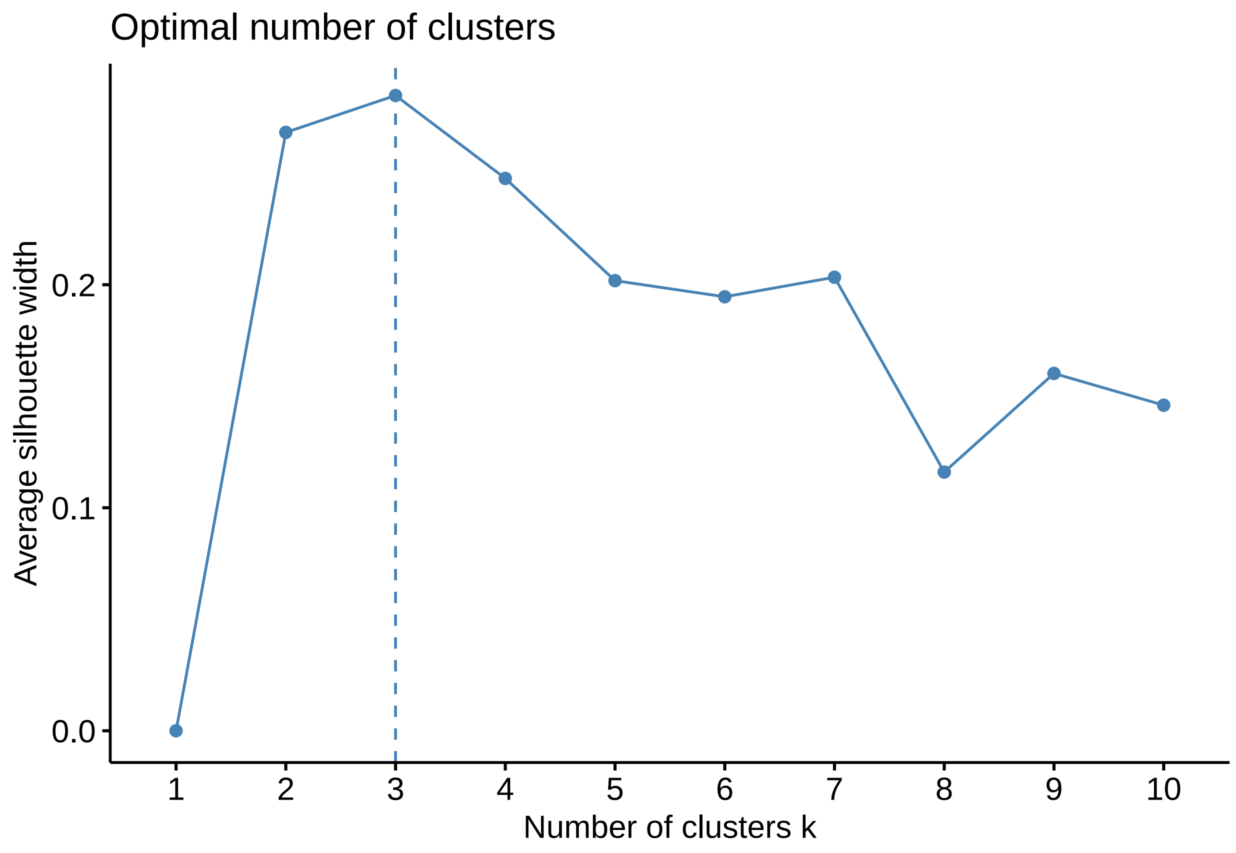
```r
set.seed(813)
factoextra::fviz_nbclust(
  x = scale_M,
  FUNcluster = kmeans,
  method = "wss"
)
```



We can see the elbow points at 2 & 3. So, probably K = 3 or 2 would be giving us ideal clusters. Further we can observe the silhouette plot to find the best k value.

To find the optimal number of clusters we can examine the silhouette plot below:

```
set.seed(813)
factoextra::fviz_nbclust(
  x = scale_M,
  FUNcluster = kmeans,
  method = "silhouette"
)
```

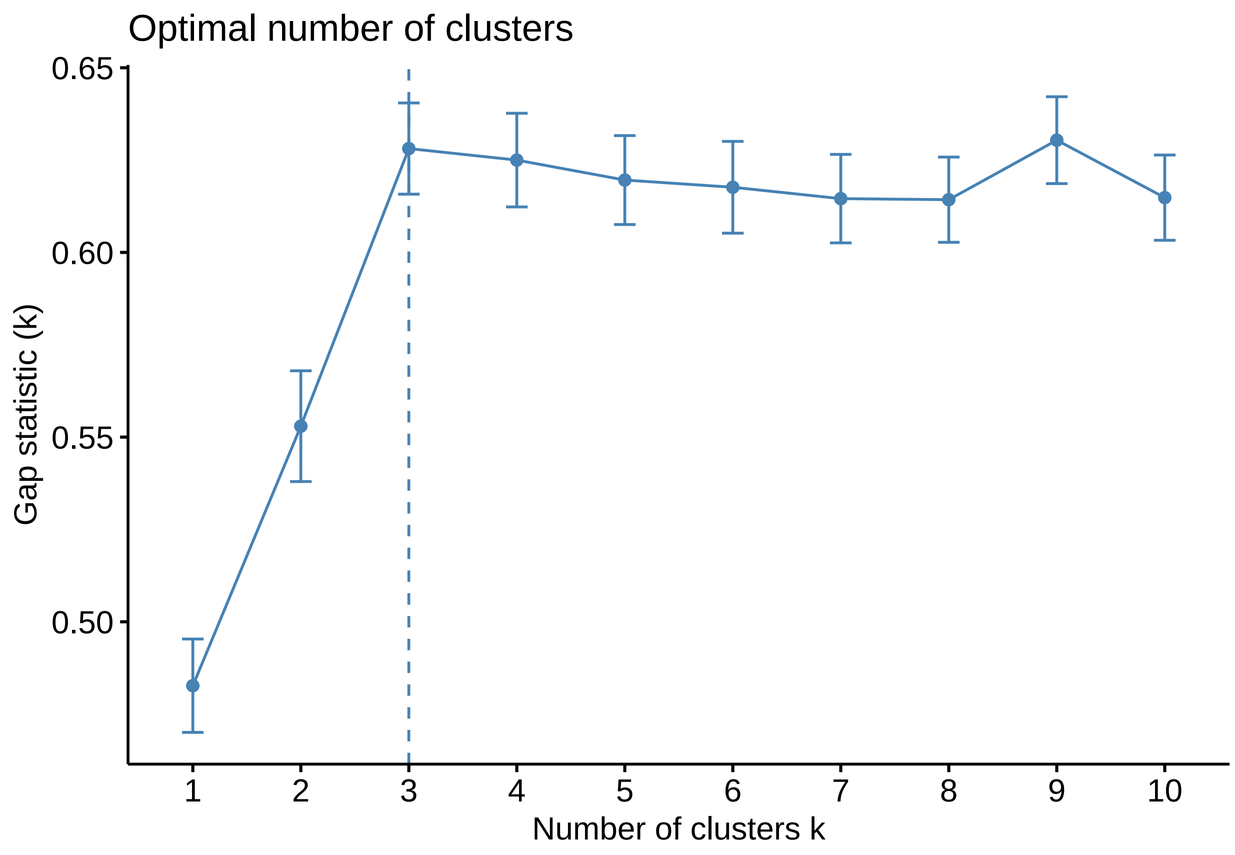


From the silhouette plot we can say that we can get ideal number of clusters when K=3.

Finding the optimal K (number of clusters) value using gap statistics:

```
set.seed(813)
clusGap_kmeans <- cluster::clusGap(
  x = scale_M,
  FUNcluster = kmeans,
  K.max = 10
)

factoextra::fviz_gap_stat(
  gap_stat = clusGap_kmeans,
)
```



From the gap statistics we can see that the optimal value of K is 3.

k-means clustering

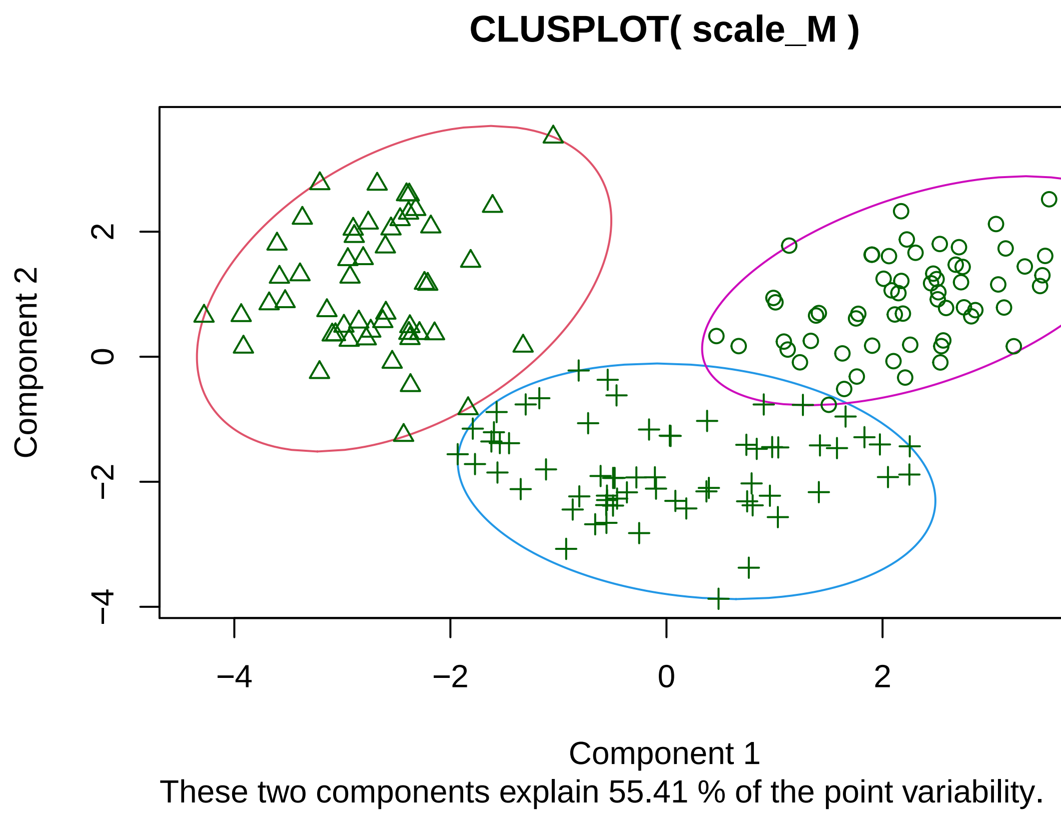
k-means clustering aims to partition the points into k groups such that the sum of squares from points to the assigned cluster centers is minimized. It takes number of groups, or initial group centers, then labels observations into groups that are close to common group centers. Then recalculates the centers and repeats.

```
kmeans_M1 <- kmeans(
  x = scale_M,
  centers = 3
)
kmeans_M1
```

```
## K-means clustering with 3 clusters of sizes 62, 51, 65
##
## Cluster means:
##      Alcohol Malic_Acid        Ash Ash_Alcanity   Magnesium Total_Phenols
## 1  0.8328826 -0.3029551  0.3636801   -0.6084749  0.57596208    0.88274724
## 2  0.1644436  0.8690954  0.1863726    0.5228924 -0.07526047   -0.97657548
## 3 -0.9234669 -0.3929331 -0.4931257    0.1701220 -0.49032869   -0.07576891
##    Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity        Hue
## 1  0.97506900          -0.56050853      0.57865427       0.1705823  0.4726504
## 2 -1.21182921           0.72402116     -0.77751312       0.9388902 -1.1615122
## 3  0.02075402          -0.03343924      0.05810161      -0.8993770  0.4605046
##        OD280    Proline
## 1  0.7770551  1.1220202
## 2 -1.2887761 -0.4059428
## 3  0.2700025 -0.7517257
##
## 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 2 3 3 3 3 3 3 3 3 3 3 3 1
##  [75] 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 2 3 3 1 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 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
##
## Within cluster sum of squares by cluster:
## [1] 385.6983 326.3537 558.6971
##  (between_SS / total_SS =  44.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

When we observe the kmeans clustering vector, we can clearly see that the dataset has been well clustered into three groups and the cluster means of each feature tell that all features are substantially influencing the output clusters (due to scaling).

```
cluster::clusplot(
  scale_M,
  kmeans_M1$cluster,
  color=TRUE,
  shade=FALSE,
  lines=0
)
```

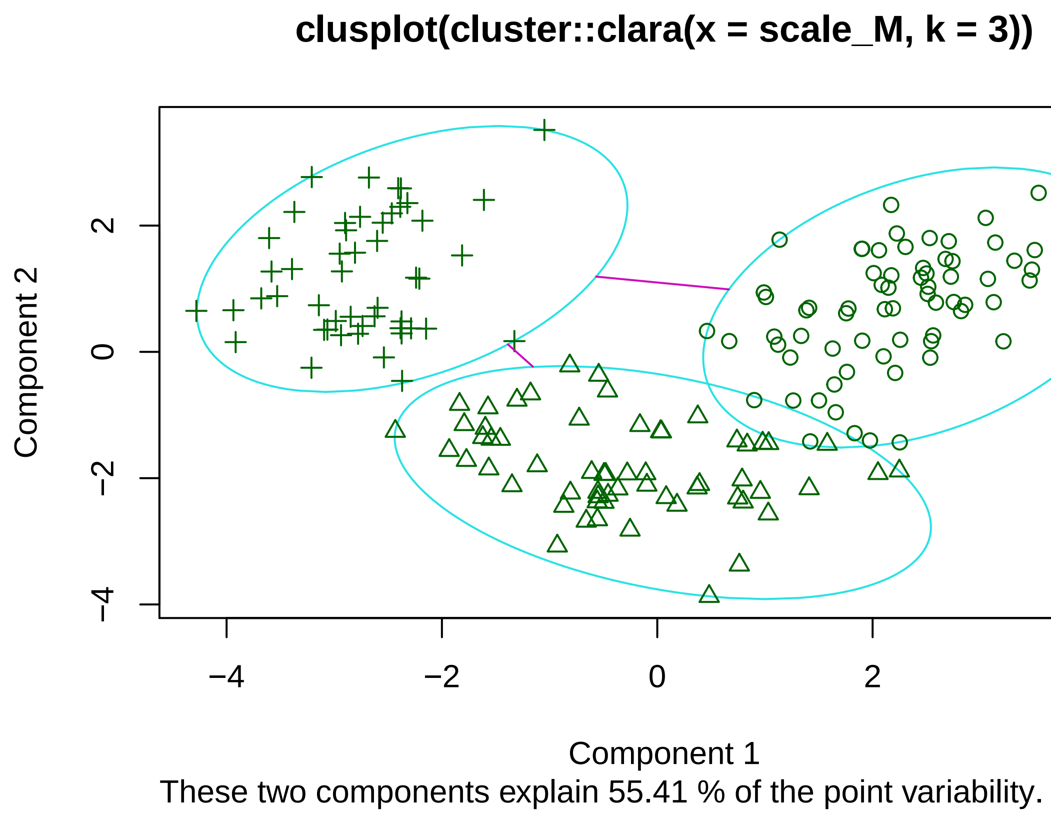


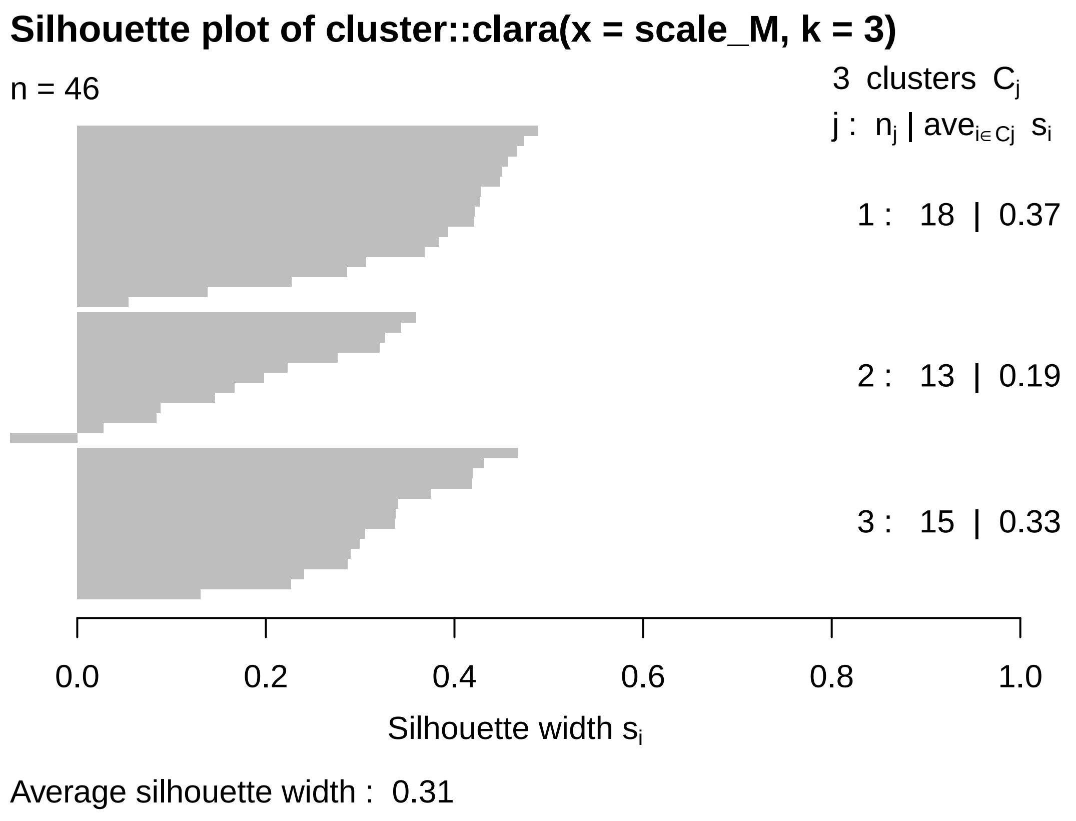
The clusters formed using kmeans() look almost perfect. The dataset is well divided into 3 clusters in the custplot of kmeans.

Partitioning using Clara method

The cluster::clara() function is a good partitioning method for large data when robustness is not needed.

```
clara_M <- cluster::clara(
  x = scale_M,
  k = 3
)
plot(clara_M)
```


clusplot(cluster::clara(x = scale_M, k = 3))
Component 2
Component 1
These two components explain 55.41 % of the point variability.
−4
−2
0
2
4



When we look at the clustplot obtained from clara, we can see that there are 3 clusters created and the clusters created look good as the boundaries of these clusters are able to partition the clusters well.

But there is a slight overlap between two of the clusters which appears to be more than the overlap that is observed in clusters made from kmeans.

Results:

```
print(clara_M)
```

```
## Call:     cluster::clara(x = scale_M, k = 3)
## Medoids:
##         Alcohol Malic_Acid         Ash Ash_Alcanity  Magnesium Total_Phenols
## [1,]  1.3542080 -0.2831754  0.12204803   -0.2080942  0.2281415     0.7268305
## [2,] -0.9246039 -0.5427655 -0.89856839   -0.1482061 -1.3822227    -1.0307762
## [3,]  0.3934117  0.8088930  0.04914686    0.6003946 -0.5420327    -0.5833854
##         Flavanoids Nonflavanoid_Phenols Proanthocyanins Color_Intensity
## [1,]  0.8917481025          -0.33630220      1.37868246       0.4925666
## [2,]  0.0007311716           0.06545479      0.06831575      -0.7152224
## [3,] -1.2707199546           0.70826598     -0.59560339       1.4501706
##             Hue      OD280    Proline
## [1,]  0.4924084  0.1948119  0.9942817
## [2,]  0.1861586  0.7863692 -0.7522631
## [3,] -1.7825902 -1.3967588 -0.3076880
## Objective function:   2.821054
## Clustering vector:    int [1:178] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
## Cluster sizes:            69 60 49
## Best sample:
##  [1]   3   7  11  13  17  20  24  30  36  37  39  47  49  50  51  55  56  62  71
## [20]  77  85  95 102 107 108 115 116 120 122 124 129 133 136 137 142 147 149 152
## [39] 154 155 157 159 161 163 164 177
##
## Available components:
##  [1] "sample"     "medoids"    "i.med"      "clustering" "objective"
##  [6] "clusinfo"   "diss"       "call"       "silinfo"    "data"
```

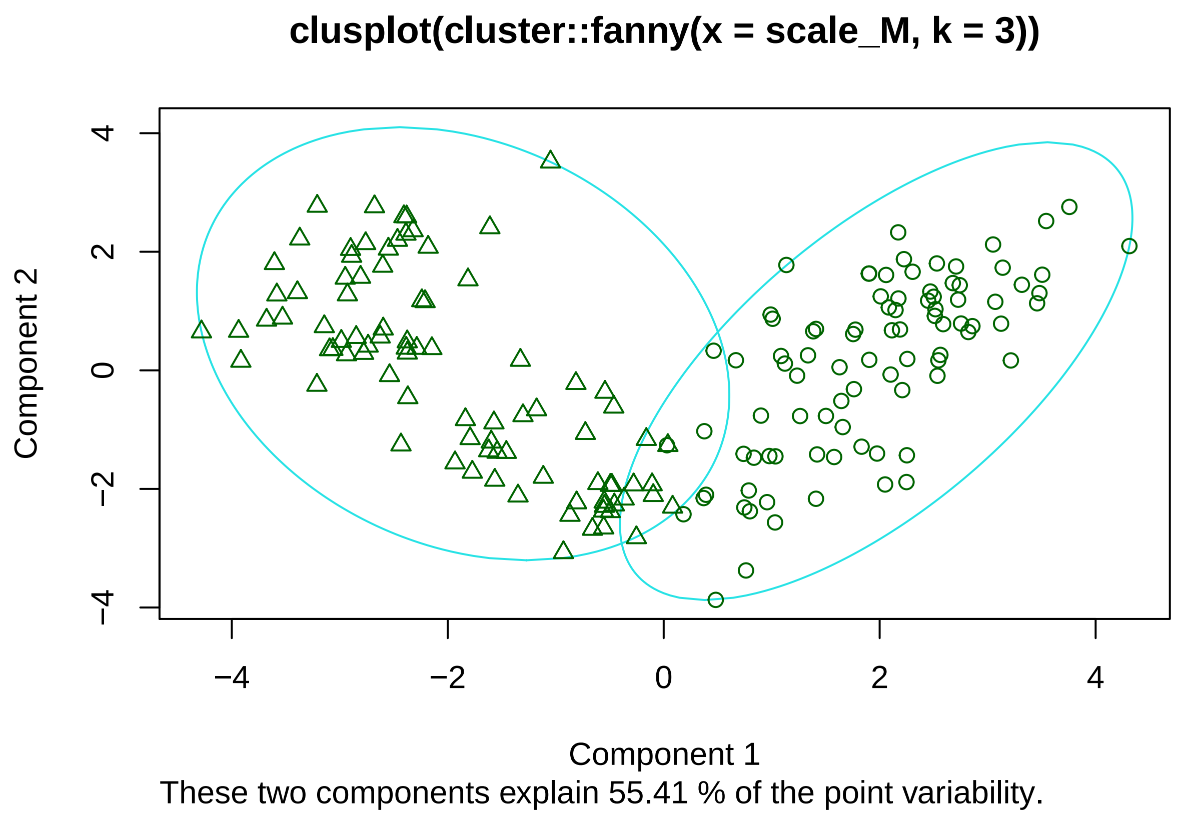
Partitioning using fanny method

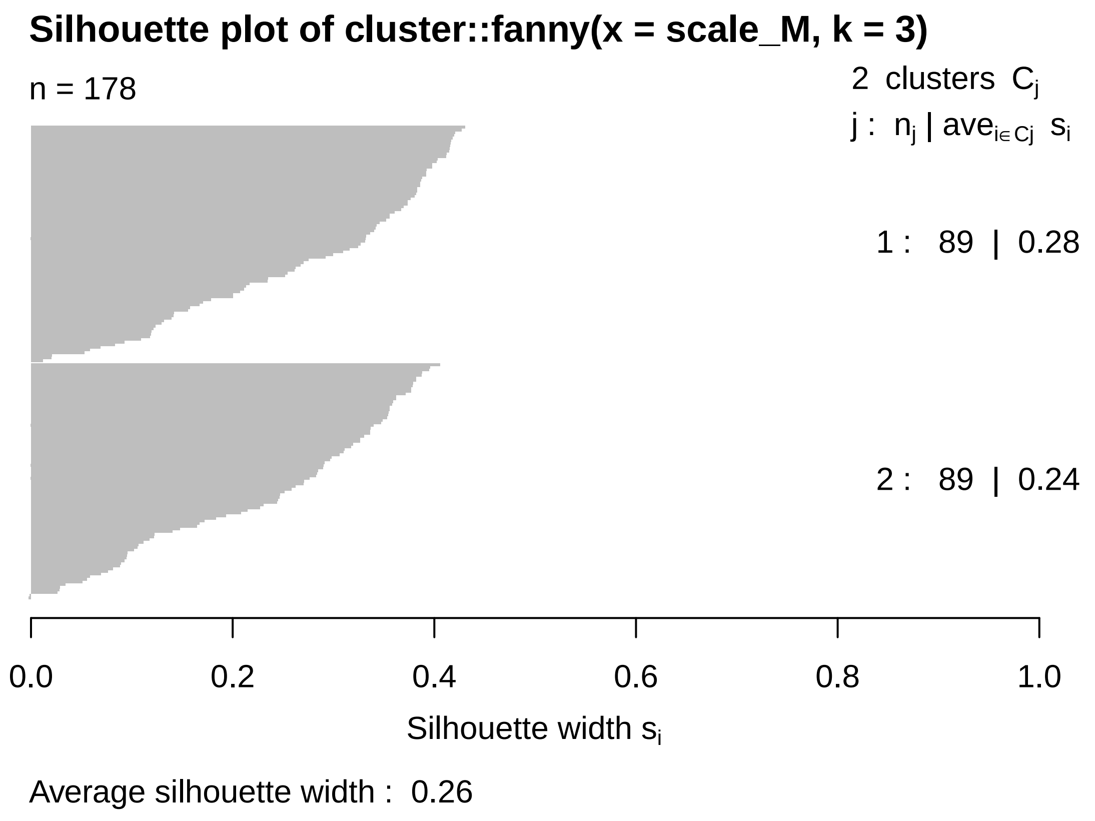
The cluster::fanny() function gives a likelihood of a point belonging to a cluster.

```
fanny_M <- cluster::fanny(
  x = scale_M,
  k = 3
)
```

```
## Warning in cluster::fanny(x = scale_M, k = 3): the memberships are all very
## close to 1/k. Maybe decrease 'memb.exp' ?
```

```
plot(fanny_M)
```





When we observe the clustplot created by fanny method, we can see that the two clusters formed are overlapping each other.However the silhouette plot looks good here suggesting that the two clusters are almost perfect. But looking at the overall picture we can say that clusters created by clara & kmeans look more promising than the clusters created using fanny.

Results:

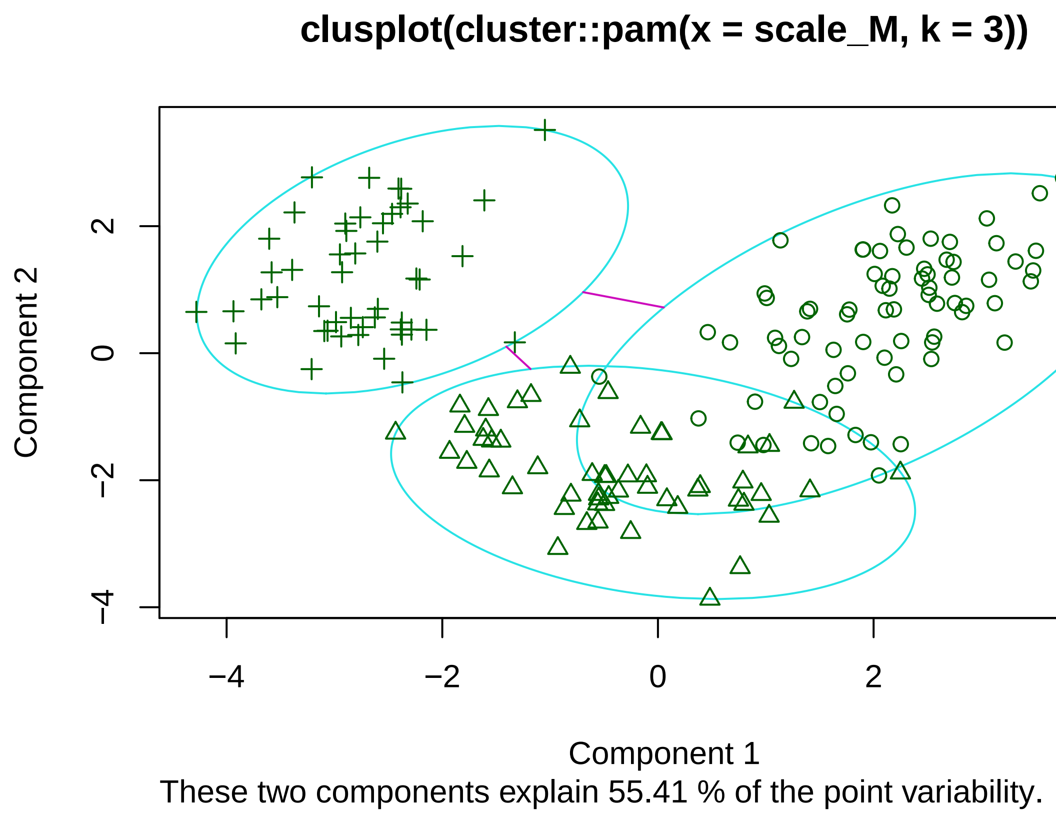
```
print(fanny_M)
```

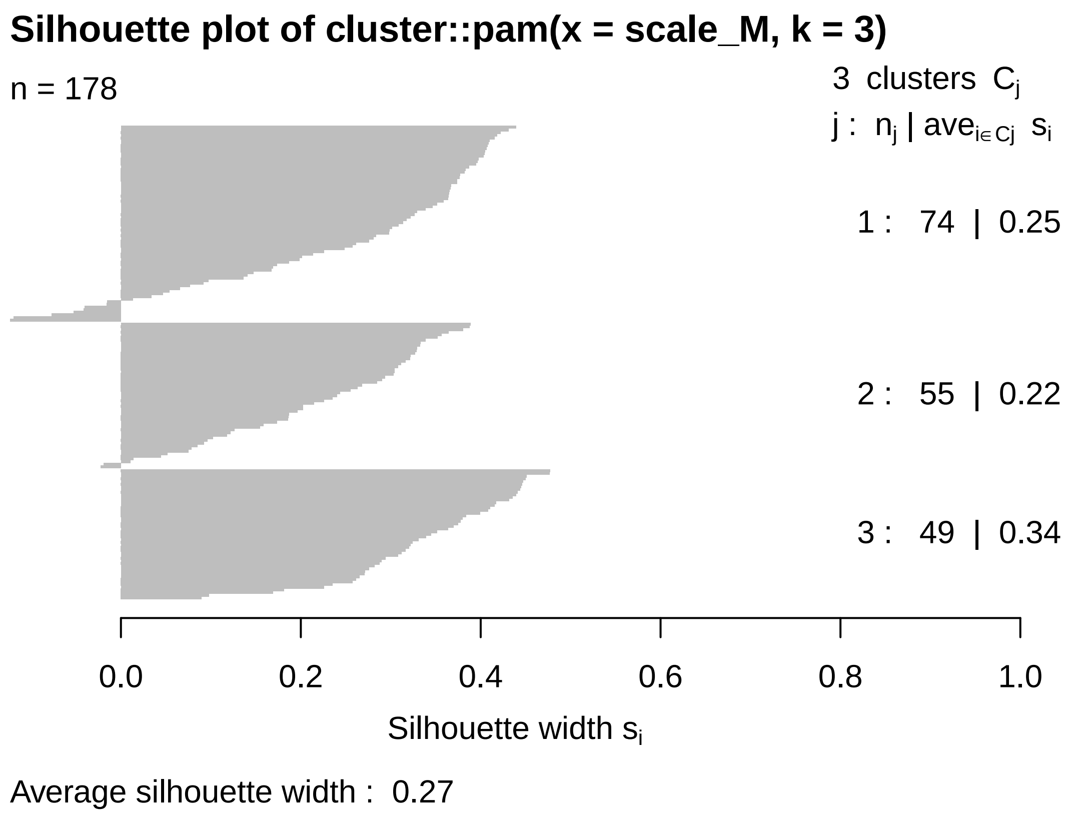
```
## Fuzzy Clustering object of class 'fanny' :
## m.ship.expon.        2
## objective     144.3284
## tolerance        1e-15
## iterations          37
## converged            1
## maxit              500
## n                  178
## Membership coefficients (in %, rounded):
##        [,1] [,2] [,3]
##   [1,]   33   33   33
##   [2,]   33   33   33
##   [3,]   33   33   33
##   [4,]   33   33   33
##   [5,]   33   33   33
##   [6,]   33   33   33
##   [7,]   33   33   33
##  [8,]   33   33   33
##  [9,]   33   33   33
## [10,]   33   33   33
## [11,]   33   33   33
## [12,]   33   33   33
## [13,]   33   33   33
## [14,]   33   33   33
## [15,]   33   33   33
## [16,]   33   33   33
## [17,]   33   33   33
## [18,]   33   33   33
## [19,]   33   33   33
## [20,]   33   33   33
## [21,]   33   33   33
## [22,]   33   33   33
## [23,]   33   33   33
## [24,]   33   33   33
## [25,]   33   33   33
## [26,]   33   33   33
## [27,]   33   33   33
## [28,]   33   33   33
## [29,]   33   33   33
## [30,]   33   33   33
## [31,]   33   33   33
## [32,]   33   33   33
## [33,]   33   33   33
## [34,]   33   33   33
## [35,]   33   33   33
## [36,]   33   33   33
## [37,]   33   33   33
## [38,]   33   33   33
## [39,]   33   33   33
## [40,]   33   33   33
## [41,]   33   33   33
## [42,]   33   33   33
## [43,]   33   33   33
## [44,]   33   33   33
## [45,]   33   33   33
## [46,]   33   33   33
## [47,]   33   33   33
## [48,]   33   33   33
## [49,]   33   33   33
## [50,]   33   33   33
## [51,]   33   33   33
## [52,]   33   33   33
## [53,]   33   33   33
## [54,]   33   33   33
## [55,]   33   33   33
## [56,]   33   33   33
## [57,]   33   33   33
## [58,]   33   33   33
## [59,]   33   33   33
## [60,]   33   33   33
## [61,]   33   33   33
##  [62,]   33   33   33
##  [63,]   33   33   33
##  [64,]   33   33   33
##  [65,]   33   33   33
##  [66,]   33   33   33
##  [67,]   33   33   33
##  [68,]   33   33   33
##  [69,]   33   33   33
##  [70,]   33   33   33
##  [71,]   33   33   33
##  [72,]   33   33   33
##  [73,]   33   33   33
##  [74,]   33   33   33
##  [75,]   33   33   33
##  [76,]   33   33   33
##  [77,]   33   33   33
##  [78,]   33   33   33
##  [79,]   33   33   33
##  [80,]   33   33   33
##  [81,]   33   33   33
##  [82,]   33   33   33
##  [83,]   33   33   33
##  [84,]   33   33   33
##  [85,]   33   33   33
##  [86,]   33   33   33
##  [87,]   33   33   33
##  [88,]   33   33   33
##  [89,]   33   33   33
##  [90,]   33   33   33
##  [91,]   33   33   33
##  [92,]   33   33   33
##  [93,]   33   33   33
##  [94,]   33   33   33
##  [95,]   33   33   33
##  [96,]   33   33   33
##  [97,]   33   33   33
##  [98,]   33   33   33
##  [99,]   33   33   33
## [100,]   33   33   33
## [101,]   33   33   33
## [102,]   33   33   33
## [103,]   33   33   33
## [104,]   33   33   33
## [105,]   33   33   33
## [106,]   33   33   33
## [107,]   33   33   33
## [108,]   33   33   33
## [109,]   33   33   33
## [110,]   33   33   33
## [111,]   33   33   33
## [112,]   33   33   33
## [113,]   33   33   33
## [114,]   33   33   33
## [115,]   33   33   33
## [116,]   33   33   33
## [117,]   33   33   33
## [118,]   33   33   33
## [119,]   33   33   33
## [120,]   33   33   33
## [121,]   33   33   33
## [122,]   33   33   33
## [123,]   33   33   33
## [124,]   33   33   33
## [125,]   33   33   33
## [126,]   33   33   33
## [127,]   33   33   33
## [128,]   33   33   33
## [129,]   33   33   33
## [130,]   33   33   33
## [131,]   33   33   33
## [132,]   33   33   33
## [133,]   33   33   33
## [134,]   33   33   33
## [135,]   33   33   33
## [136,]   33   33   33
## [137,]   33   33   33
## [138,]   33   33   33
## [139,]   33   33   33
## [140,]   33   33   33
## [141,]   33   33   33
## [142,]   33   33   33
## [143,]   33   33   33
## [144,]   33   33   33
## [145,]   33   33   33
## [146,]   33   33   33
## [147,]   33   33   33
## [148,]   33   33   33
## [149,]   33   33   33
## [150,]   33   33   33
## [151,]   33   33   33
## [152,]   33   33   33
## [153,]   33   33   33
## [154,]   33   33   33
## [155,]   33   33   33
## [156,]   33   33   33
## [157,]   33   33   33
## [158,]   33   33   33
## [159,]   33   33   33
## [160,]   33   33   33
## [161,]   33   33   33
## [162,]   33   33   33
## [163,]   33   33   33
## [164,]   33   33   33
## [165,]   33   33   33
## [166,]   33   33   33
## [167,]   33   33   33
## [168,]   33   33   33
## [169,]   33   33   33
## [170,]   33   33   33
## [171,]   33   33   33
## [172,]   33   33   33
## [173,]   33   33   33
## [174,]   33   33   33
## [175,]   33   33   33
## [176,]   33   33   33
## [177,]   33   33   33
## [178,]   33   33   33
## Fuzzyness coefficients:
##   dunn_coeff   normalized
## 3.333333e-01 2.664535e-15
## Closest hard clustering:
##   [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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 1 1 2 1 1 1 2 1 2 1 2 1
##  [75] 1 2 1 2 1 1 1 1 2 2 1 1 2 2 2 2 2 2 2 1 1 1 2 1 1 1 1 2 2 2 1 2 2 2 2 1 1
## [112] 2 2 2 2 1 2 2 2 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 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
## k_crisp (= 2) < k !!
##
## Available components:
##  [1] "membership"  "coeff"       "memb.exp"    "clustering"  "k.crisp"
##  [6] "objective"   "convergence" "diss"        "call"        "silinfo"
## [11] "data"
```

Partitioning using pam method

The cluster::pam() algorithm is the robust version of k-means. It uses medoids, and centers are observations in the data set.

```
pam_M <- cluster::pam(
  x = scale_M,
  k = 3
)
plot(pam_M)
```


clusplot(cluster::pam(x = scale_M, k = 3))
Component 2
2
0
−2
−4
−4
−2
0
2
4
Component 1
These two components explain 55.41 % of the point variability.



We can see that there is significant overlap between the clusters formed by pam(). Kmeans, clara & fanny are giving better results compared to pam().

Results:

```
print(pam_M)
```

```
## Medoids:
##       ID    Alcohol Malic_Acid         Ash Ash_Alcanity   Magnesium
## [1,]  36  0.5904981 -0.4711544  0.15849862    0.3009543  0.01809398
## [2,] 107 -0.9246039 -0.5427655 -0.89856839   -0.1482061 -1.38222271
## [3,] 149  0.3934117  0.8088930  0.04914686    0.6003946 -0.54203270
##      Total_Phenols    Flavanoids Nonflavanoid_Phenols Proanthocyanins
## [1,]     0.6469393  0.9518166597          -0.81841060      0.47016154
## [2,]    -1.0307762  0.0007311716           0.06545479      0.06831575
## [3,]    -0.5833854 -1.2707199546           0.70826598     -0.59560339
##      Color_Intensity        Hue      OD280    Proline
## [1,]      0.01807806  0.3611585  1.2089101  0.5497067
## [2,]     -0.71522236  0.1861586  0.7863692 -0.7522631
## [3,]      1.45017064 -1.7825902 -1.3967588 -0.3076880
## 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 2 2 2 1 2 1 2 1
##  [75] 1 2 2 2 2 1 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 1 1 2 1 2 2 2 2 2 2 2 2 2 2 1 1
## [112] 2 2 2 2 2 2 2 2 2 1 1 2 2 1 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 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
## 2.910808 2.806293
##
## Available components:
##  [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
##  [6] "clusinfo"   "silinfo"    "diss"       "call"       "data"
```

Final Thoughts on Partitioning Algorithm below are the key observations from partitioning algorithm:

- The best value for number of clusters (K) is 3 which is seen from the plots created using elbow method, Silhouette calculations and gap statistics.
- The best model for partitioning algorithm is given by kmeans() as the clusters are well separated from each other.

Hierarchical Clustering

sample of distance matrix:

```
dist(
  x = scale_M[1:5,]
)
```

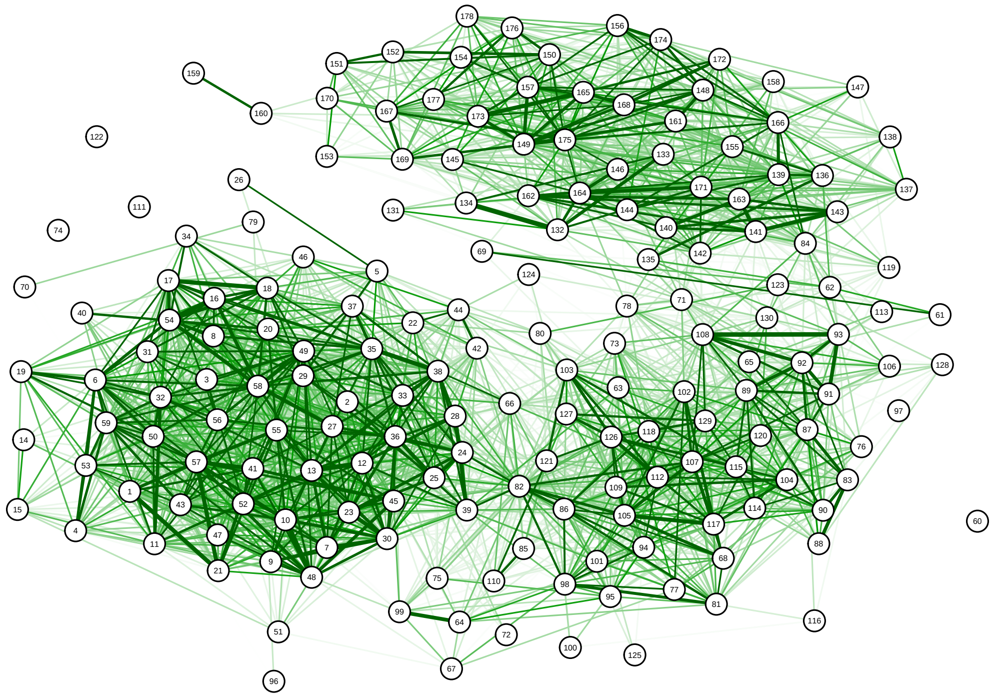
```
##          1        2        3        4
## 2 3.487697
## 3 3.018094 4.131258
## 4 2.834509 4.348349 3.237354
## 5 3.556821 4.614454 2.972721 4.483310
```

creating the distance matrix for hierarchical clustering:

```
dist_M <- dist(
  x = scale_M
)
```

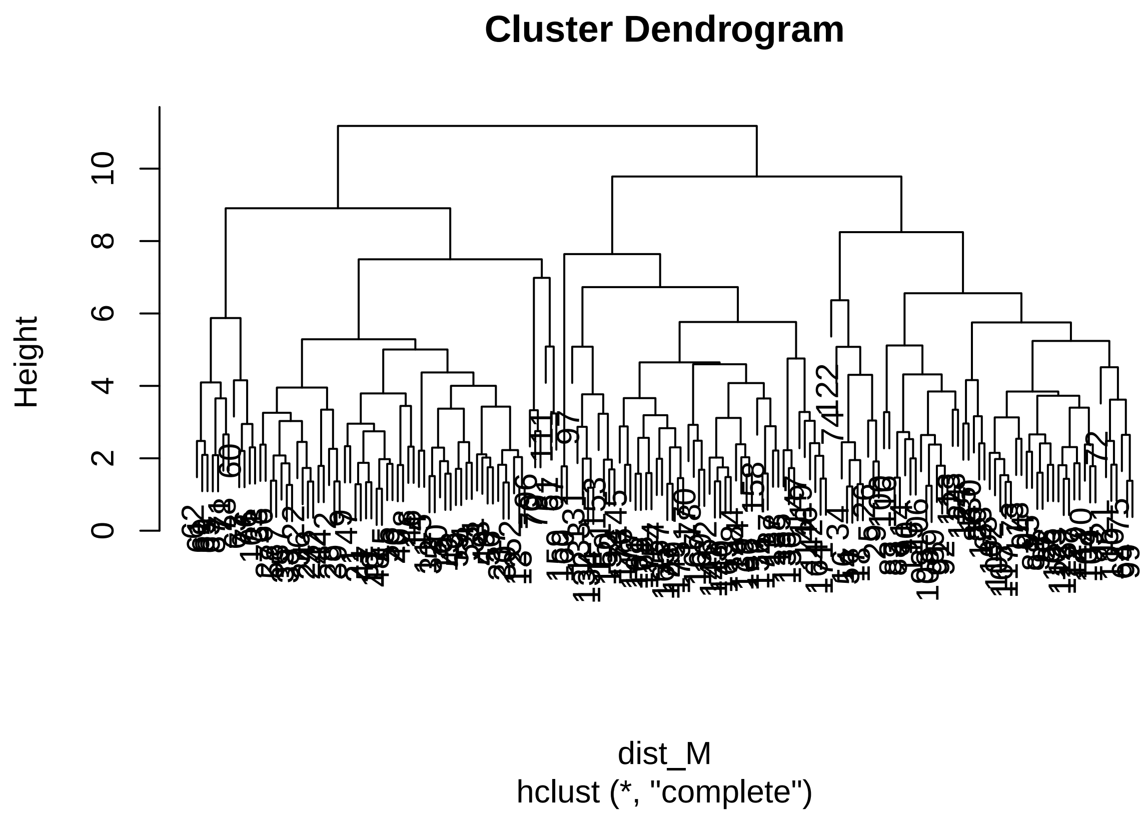
Now that we have created our distance matrix, we can plot the qgraph that would show association between rows based on thickness of the lines.

```
library("qgraph")
qgraph::qgraph(
  input = 1/dist_M,
  layout="spring",
  minimum = 0.3
)
```



From the plot it looks like the dataset can be clustered into 3 groups. Further we can use hclust() function to execute agglomerative clustering which is joining the two most similar clusters and continuing until there is just a single cluster.

```
hclust_M <- hclust(
  d = dist_M,
  method = "complete"
)
plot(
  x = hclust_M
)
```



The above dendogram shows a complete linkage method where we can see that if we cut this dendogram at a height of 8.8 (approx) we might get 3 clusters that are splitting the data into (approximately) equal sized clusters.

Lets cut the complete linkage tree so that we are able to get 3 clusters.

```
cutree_hclust_M <- cutree(
  tree = hclust_M,
  k = 3
)
```

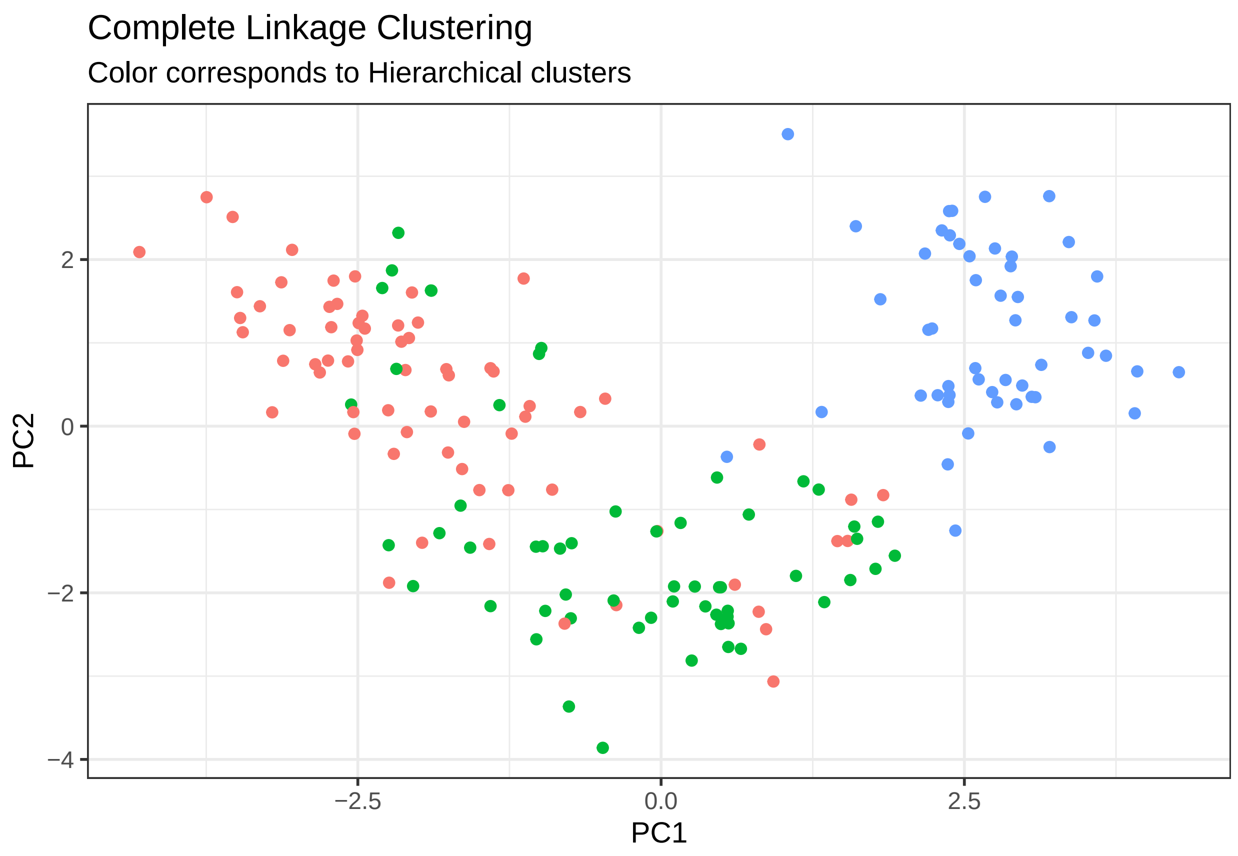
Plotting the clusters created by hierarchical clustering using complete linkage method:

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
prcomp_M <- data.frame(
  prcomp(
    x = scale_M,
    center = FALSE,
    scale. = FALSE
  )$x[,1:2],
  Cluster = as.character(cutree_hclust_M),
  stringsAsFactors = FALSE
)
```

```r
ggplot(prcomp_M) +
  aes(x = PC1,y = PC2,color = Cluster,fill = Cluster,group = Cluster) +
  geom_point() +
  ggtitle("Complete Linkage Clustering","Color corresponds to Hierarchical clusters") +
  theme_bw() +
  theme(legend.position = "none")
```



Finding the coefficient of a hierarchical clustering using complete linkage method:

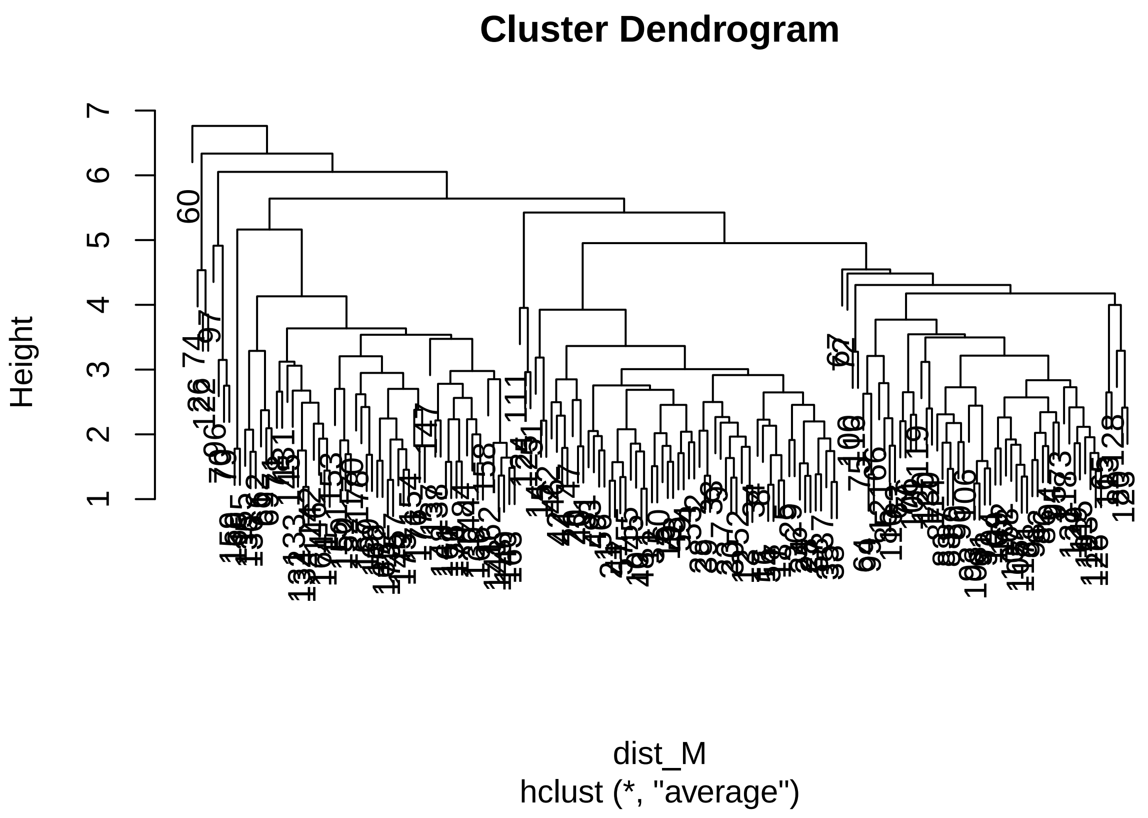
```r
cluster::coef.hclust(hclust_M)
```

```
## [1] 0.815931
```

coefficient of 0.815 for hierarchical clustering using complete linkage method suggests that the hierarchical clustering is doing a better job of creating clusters with approximately equal number of observations. This is something we see from the cluster plot created earlier as well.

Further we can check the dendogram for average linkage method:

```r
hclust_average_M <- hclust(
  d = dist_M,
  method = "average"
)
plot(
  x = hclust_average_M
)
```



From the average linkage dendogram we can see that record 60 appears on a different branch of the tree. This could be a outlier in the wine data set.

Finding the coefficient of a hierarchical clustering using average linkage method:

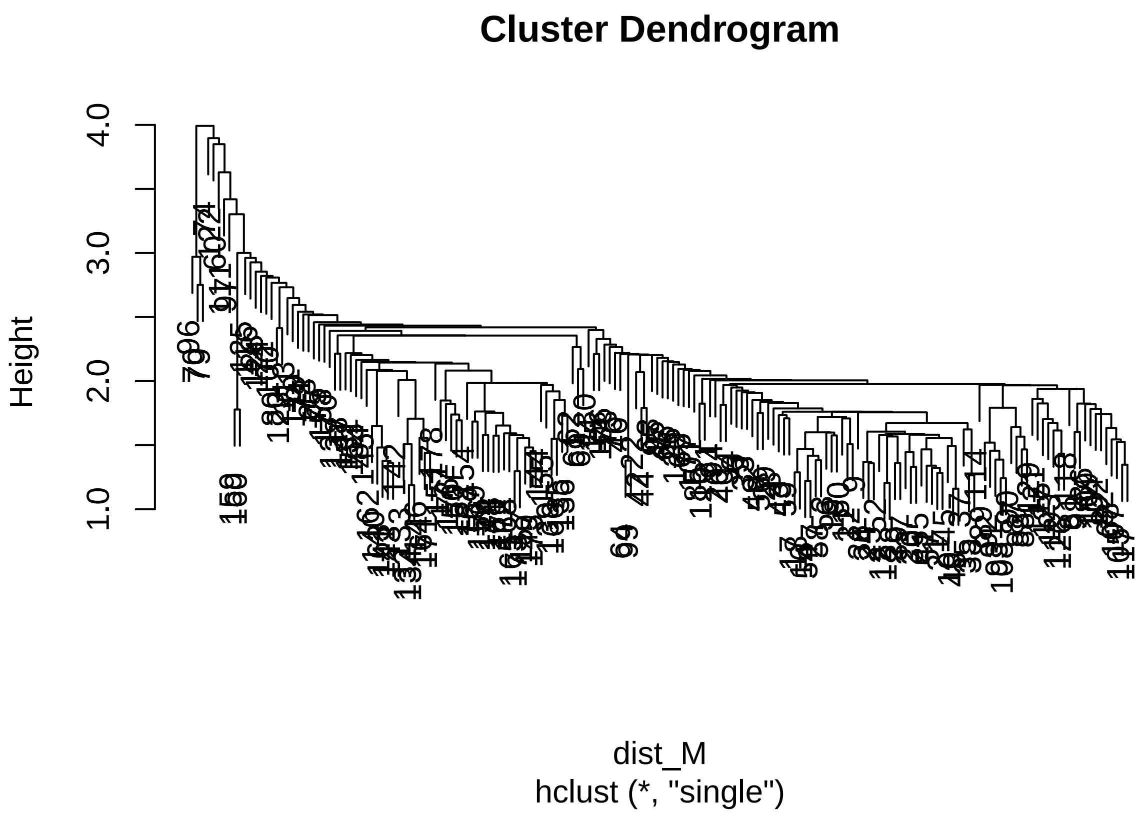
```
cluster::coef.hclust(hclust_average_M)
```

```
## [1] 0.7006964
```

This coefficient helps us to detect outliers in our dataset but we can further check hierarchical clustering using single linkage.

Checking dendogram for hierarchical clustering using single linkage method:

```
hclust_single_M <- hclust(
  d = dist_M,
  method = "single"
)
plot(
  x = hclust_single_M
)
```



Finding the coefficient of a hierarchical clustering using single linkage method:

```
cluster::coef.hclust(hclust_single_M)
```

```
## [1] 0.5379128
```

We can see the coefficient for hierarchical clustering using single linkage is low (around 0.53) which indicates that this model is better for outlier detection.

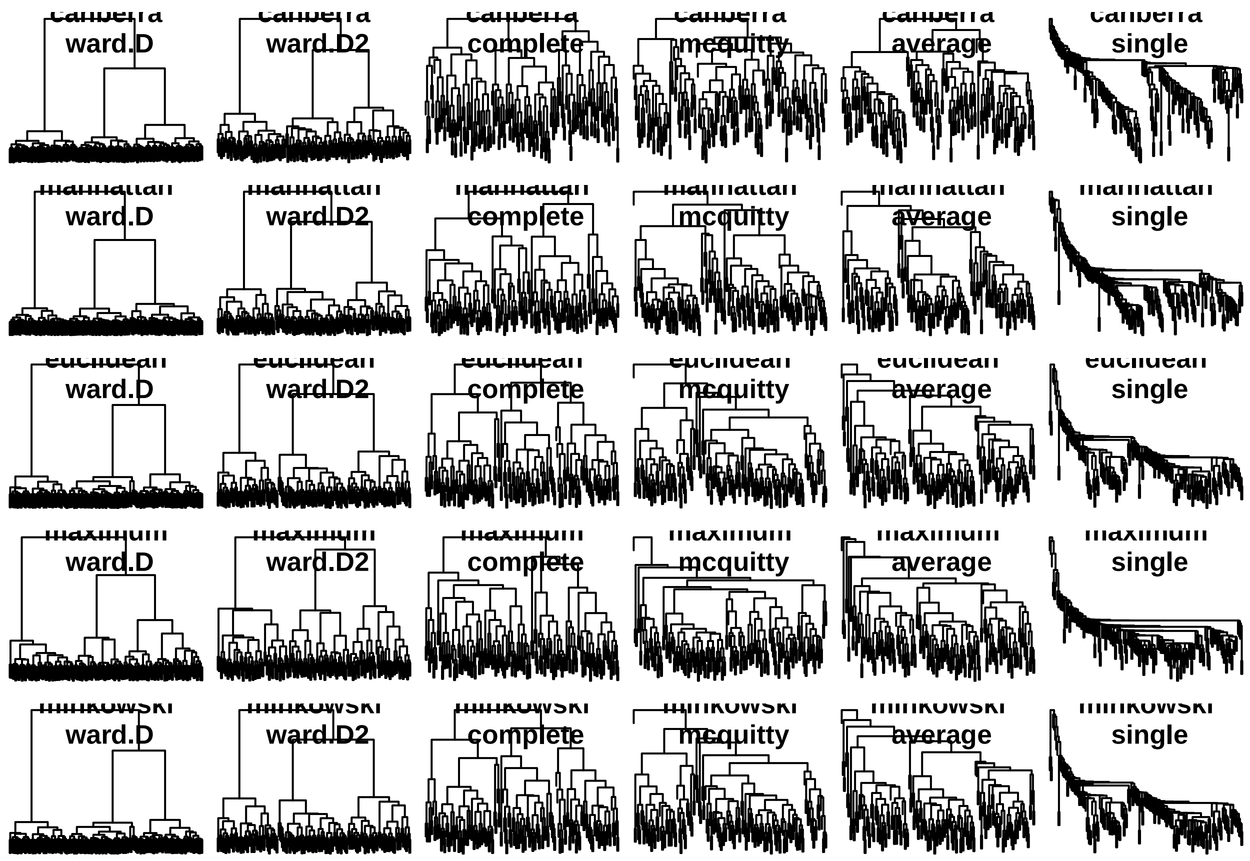
Analysis of hierarchical clustering parameters To check the best hierarchical model for outlier detection and partitioning data into (approximately) equal sized groups I have created dendograms for different combinations of distance metric and methods of hierarchical clustering. (Which are key parameters of hierarchical clustering)

First, I am creating distance metric using "canberra","manhattan","euclidean","maximum" and "minkowski" method and similarly defining different hierarchical clustering methods like "ward.D","ward.D2","complete","mcquitty","avera
and "single" linkage in v_hclust.

```
v_dist <- c(
  "canberra","manhattan","euclidean","maximum","minkowski"
)
list_dist <- lapply(
  X = v_dist,
  FUN = function(distance_method) dist(
    x = scale_M,
    method = distance_method
  )
)
names(list_dist) <- v_dist
v_hclust <- c(
  "ward.D","ward.D2","complete","mcquitty","average","single"
)
```

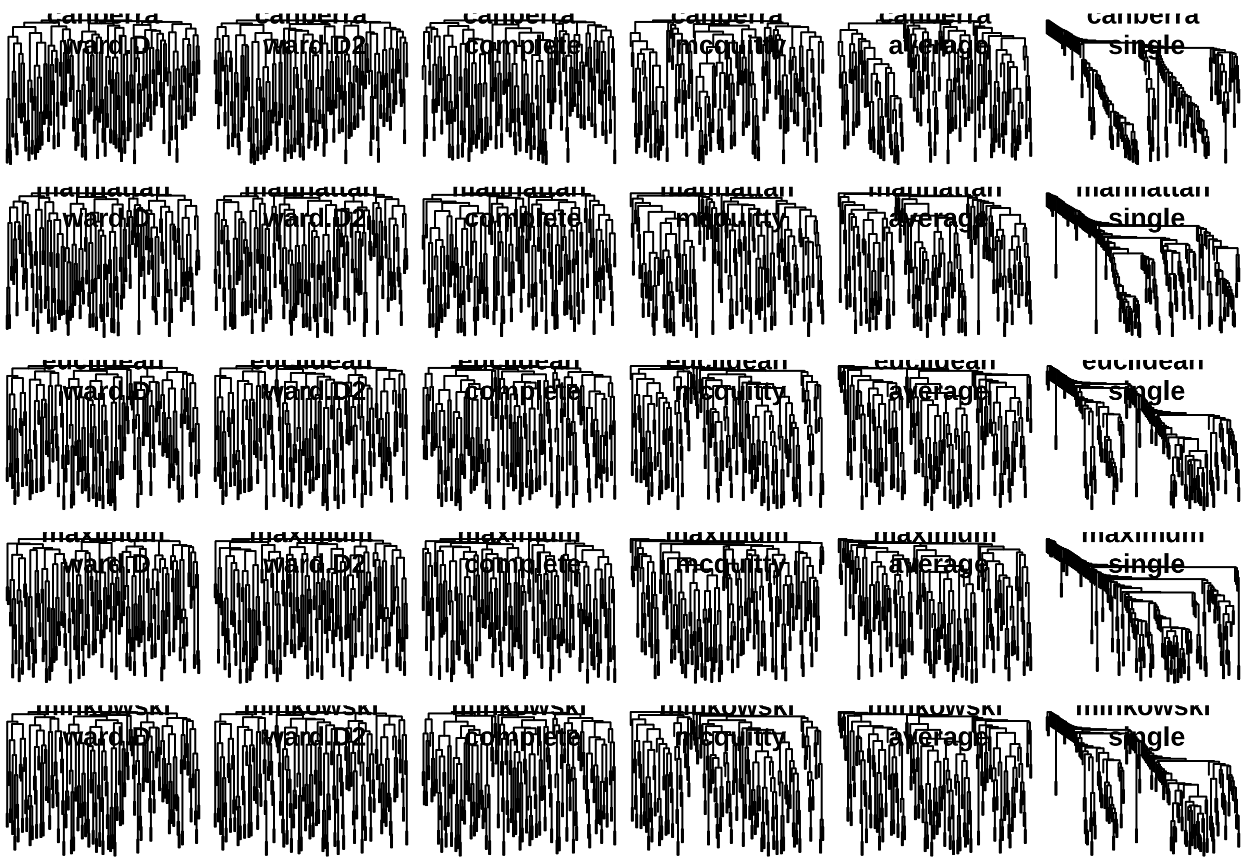
Below are different dendograms for hierarchical clustering models having different clustering methods and distance metrics:

```
list_hclust <- list()
for(j in v_dist) for(k in v_hclust) list_hclust[[j]][[k]] <- hclust(
  d = list_dist[[j]],
  method = k
)
par(
  mfrow = c(length(v_dist),length(v_hclust)),
  mar = c(0,0,0,0),
  mai = c(0,0,0,0),
  oma = c(0,0,0,0)
)
for(j in v_dist) for(k in v_hclust) plot(
  x = list_hclust[[j]][[k]],
  labels = FALSE,
  axes = FALSE,
  main = paste("\n",j,"\n",k)
)
```



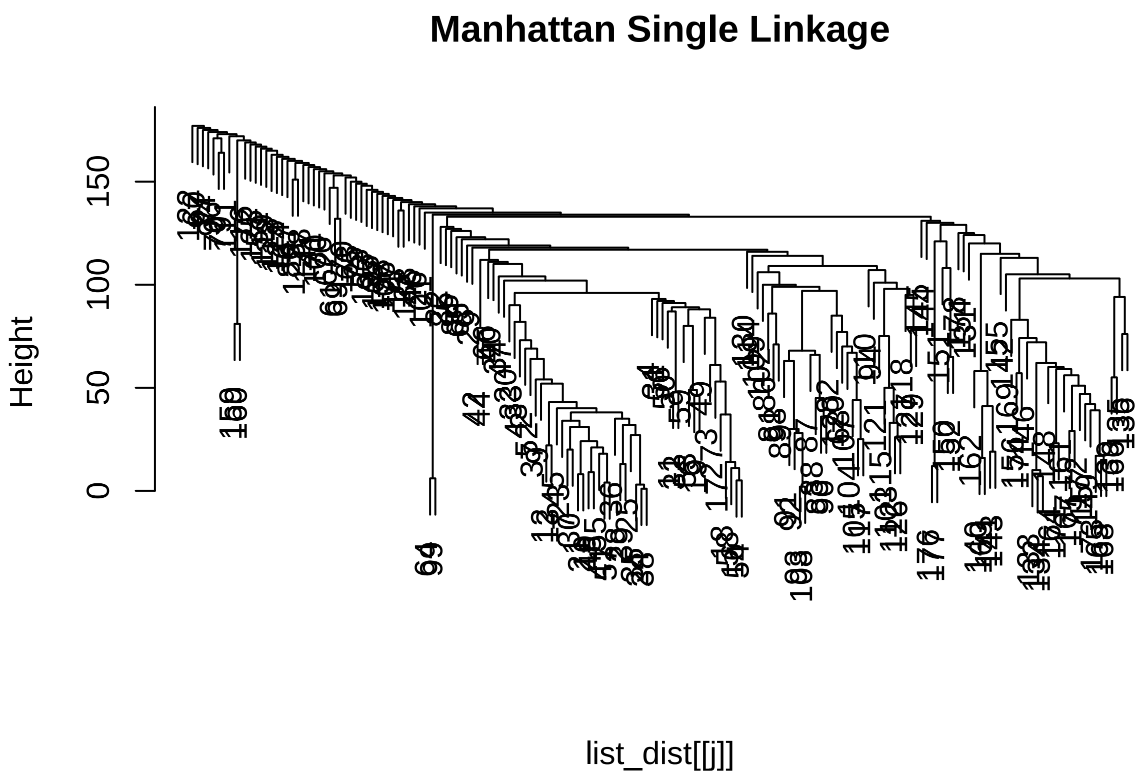
For comparison, the heights of these dendograms should be on same scale. Thus the heights of the dendograms are adjusted so that best two models can be chosen.

```r
for(j in v_dist) for(k in v_hclust) list_hclust[[j]][[k]]$height <- rank(list_hclust[[j]][[k]]$height)
par(
  mfrow = c(length(v_dist),length(v_hclust)),
  mar = c(0,0,0,0),
  mai = c(0,0,0,0),
  oma = c(0,0,0,0)
)
for(j in v_dist) for(k in v_hclust) plot(
  x = list_hclust[[j]][[k]],
  labels = FALSE,
  axes = FALSE,
  main = paste("\n",j,"\n",k)
)
```



Based on the above matrix of dendogram plots building a model that would be good for outlier detection:

```
plot(
  x = list_hclust[["manhattan"]][["single"]],
  main = "Manhattan Single Linkage",
  sub = ""
)
```



From the dendogram created using manhattan single linkage hierarchical model, we can say that rows 159 & 160 are appearding distinct from the rest of the tree brances but are highly associated to eachother. these can be potential outliers that need to be further investigated. Same is the case with observations in row 64 & 99.

Finding the clustering coefficient of this model:

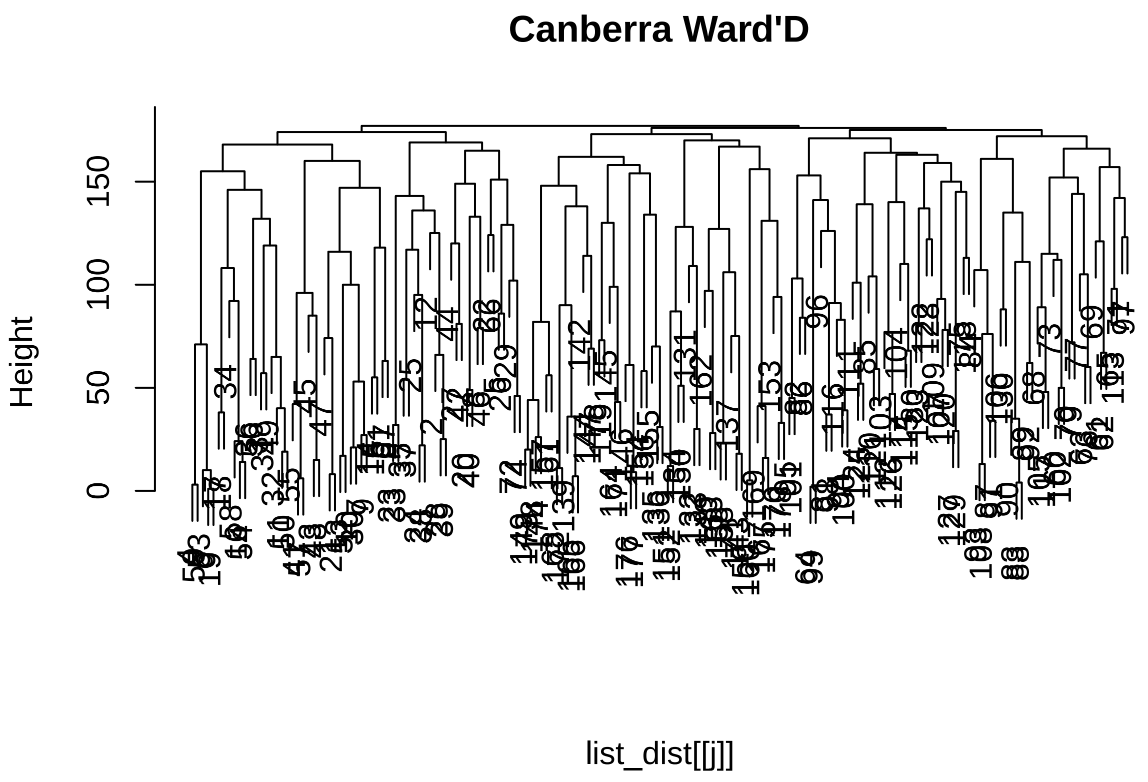
```
cluster::coef.hclust(list_hclust[["manhattan"]][["single"]])
```

```
## [1] 0.5487526
```

The clustering coefficient for the hierarchical clustering model using “single” linkage method and distance metric created by “manhattan” method is low (around 0.548) which suggest this model to be good for outlier detection.

Based on the above matrix of dendogram plots building a model that would be good for partitioning data into equal sized groups:

```
plot(
  x = list_hclust[["canberra"]][["ward.D"]],
  main = "Canberra Ward'D",
  sub = ""
)
```



From the above dendogram we can say that this model is able to partition the data into equal sized groups.

Finding the clustering coefficient of this model:

```
cluster::coef.hclust(list_hclust[["canberra"]][["ward.D"]])
```

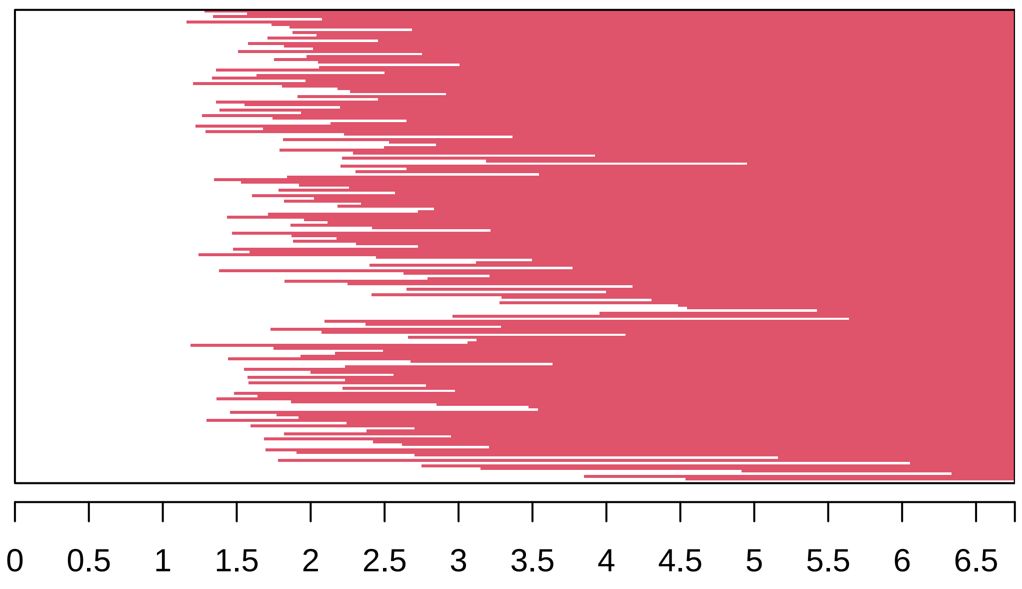
```
## [1] 0.7017394
```

The clustering coefficient for the hierarchical clustering model using "ward.D" method and distance metric created by "canberra" method is high (around 0.701) which suggest this model to be good for partitioning the data into equal sized groups.

Hierarchical clustering using agnes() method

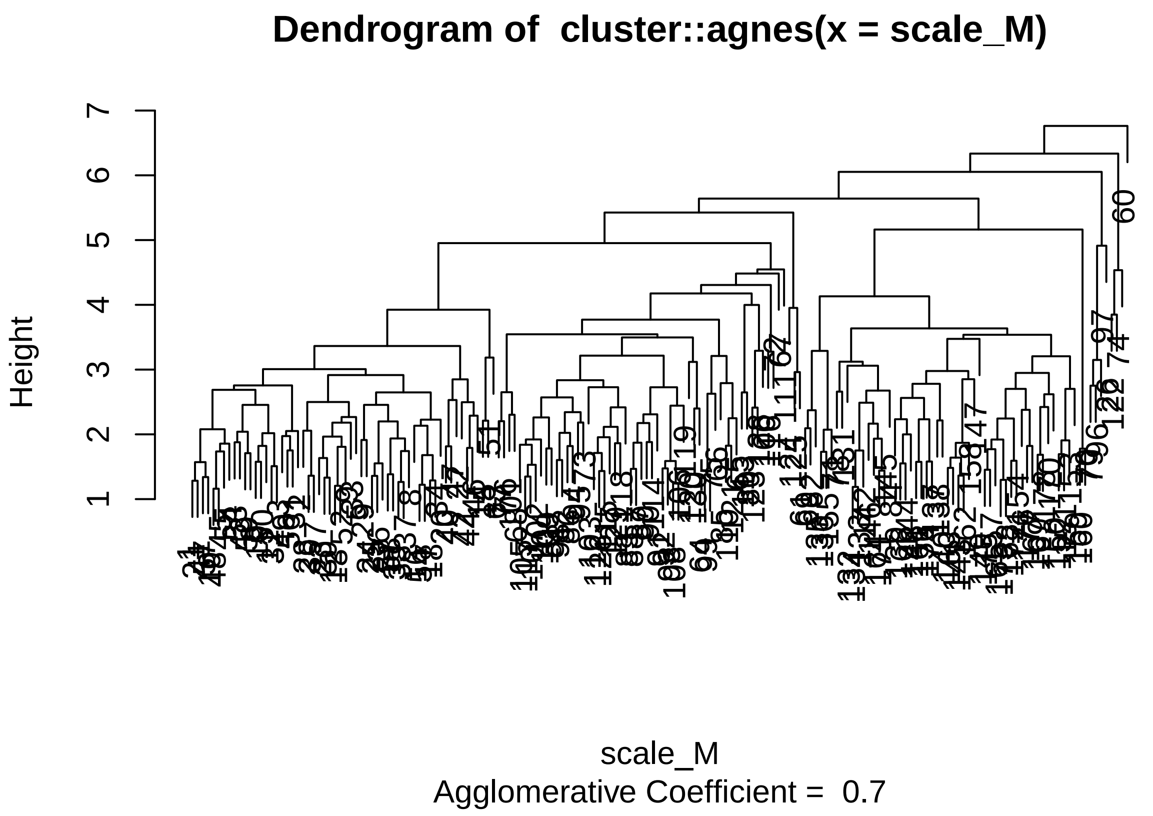
```
agnes_M <- cluster::agnes(scale_M)
plot(agnes_M)
```

Banner of cluster::agnes(x = scale_M)



Height

Agglomerative Coefficient = 0.7



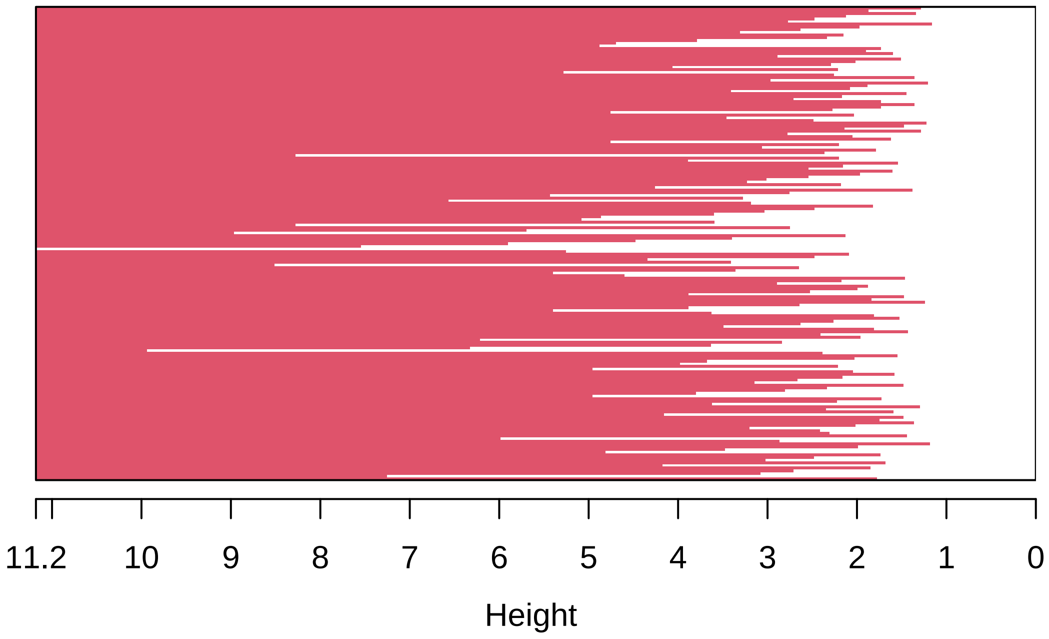
From the dendogram created by agnes() function we see that few of the records appear differently in the tree. the record of row 60 looks like an outlier in this dataset.

Thus we can say that agnes() function which uses agglomerative nesting algorithm for clustering is enabling us to detect outliers in the dataset.

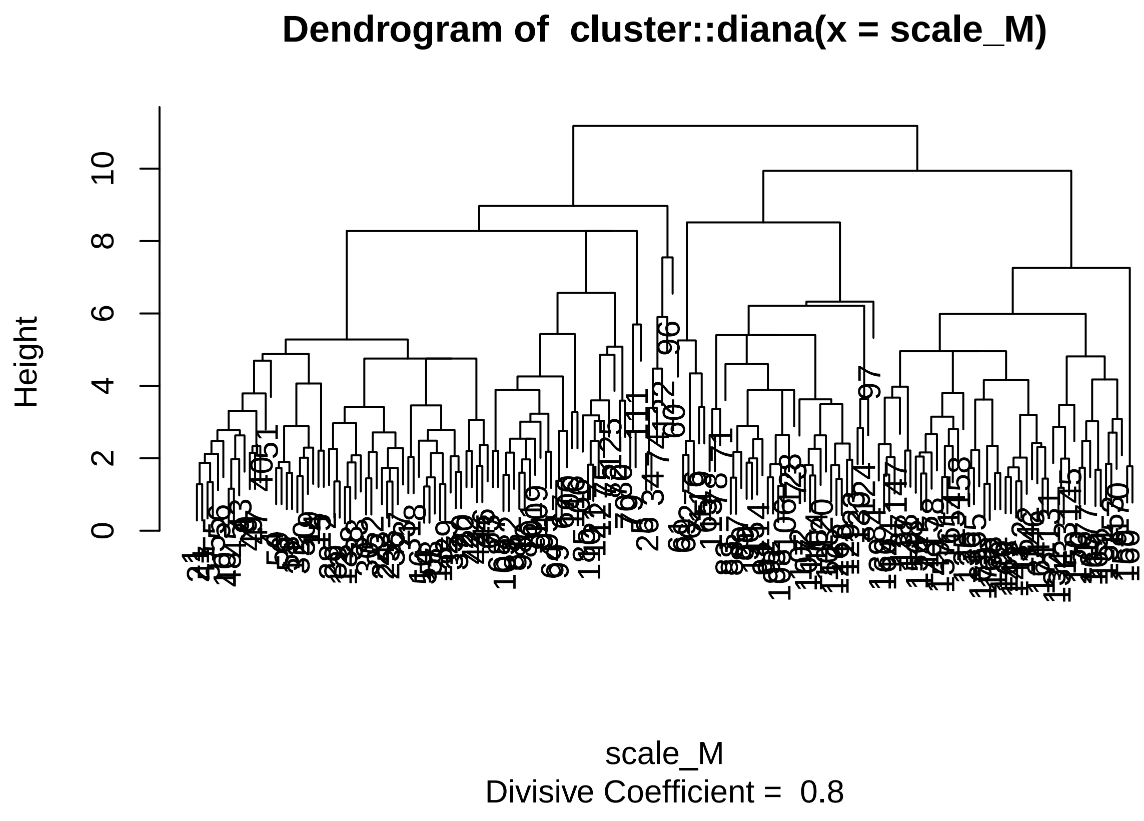
Hierarchical clustering using diana() method

```
diana_M <- cluster::diana(scale_M)
plot(diana_M)
```

Banner of cluster::diana(x = scale_M)



Divisive Coefficient = 0.8



From the dendogram created using Diana() function, we can say that the tree appears to be well partitioned. If we would cut the tree at a height between 8.7 to 8.9 (approx) we can get almost evenly partitioned clusters.

Even the coefficient of clustering for the hierarchical model created using diana() is around 0.8 which suggests that well partitioned clusters can be formed of the dataset.

Hierarchical clustering using mona() method

To use mona() function there is need to convert data into binary format.

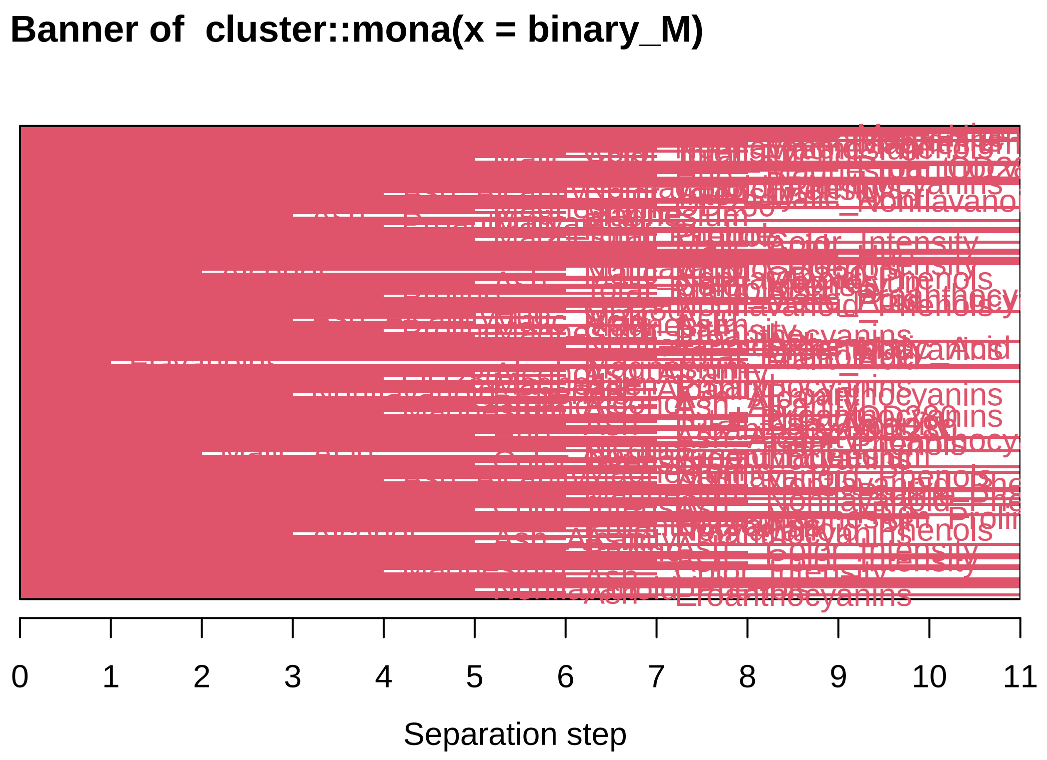
```
binary_M <- scale_M
for(j in 1:ncol(binary_M)) binary_M[,j] <- as.numeric(
  binary_M[,j] > median(binary_M[,j])
)
```

Now, as values of all variables are converted into 0’s and 1’s, monothetic analysis clustering of binary variables can be done (mona).

```
mona_M <- cluster::mona(binary_M)
print(mona_M)
```

```
## mona(x, ..) fit;  x of dimension 178x13
## Order of objects:
##    [1]   1   6  19  53  59  13  52  16  27  14  34  35  37   3  15  58  49   8
##   [19]   7   4  20  40  43  46  29  54   5  17  18  31  36  56  25  72   2  28
##   [37]  39  42   9  10  30  67  12  11  47  44  21  32  50  41  48  55  57  23
##   [55]  33  22  24  38  66  45  51  96  82 101  64  81  94  95 100  98  99  85
##  [73] 111  26  74  75 110  80 122 112 121 125 124 115 126 116 127 128 129  60
##  [91]  77 102 131  63  73  68 105 117  70  86 118  61  79  71  69  62 155  65
## [109]  83  88  92 114  87  91 107  90  93 108 104 109 135 159 160  76  78 119
## [127] 120 136 168 171 172 145 164  84 133 138 148 161 158 132 152 140 163 134
## [145]  89 130 141 137 113 103 106  97 123 139 154 157 142 156 174 143 149 173
## [163] 178 144 147 165 166 146 176 162 167 169 175 177 150 151 170 153
## Variable used:
##   [1] NULL                 NULL                 NULL
##   [4] Hue                  Magnesium            NULL
##   [7] Proanthocyanins      Magnesium            Nonflavanoid_Phenols
##  [10] Magnesium            Color_Intensity      Total_Phenols
##  [13] Malic_Acid           NULL                 NULL
##  [16] OD280                Proanthocyanins      Magnesium
##  [19] Hue                  NULL                 NULL
##  [22] NULL                 Proanthocyanins      Nonflavanoid_Phenols
##  [25] Color_Intensity      Ash_Alcanity         Color_Intensity
##  [28] Malic_Acid           Nonflavanoid_Phenols OD280
##  [31] NULL                 Magnesium            Proline
##  [34] Ash                  Magnesium            NULL
##  [37] Malic_Acid           Proanthocyanins      NULL
##  [40] NULL                 Proline              Total_Phenols
##  [43] Magnesium            NULL                 Color_Intensity
##  [46] Malic_Acid           NULL                 NULL
##  [49] Hue                  NULL                 NULL
##  [52] NULL                 Color_Intensity      Nonflavanoid_Phenols
##  [55] Alcohol              Malic_Acid           OD280
##  [58] Nonflavanoid_Phenols Ash                  Magnesium
##  [61] Color_Intensity      Total_Phenols        Malic_Acid
##  [64] Proline              Proanthocyanins      Malic_Acid
##  [67] Proanthocyanins      Nonflavanoid_Phenols OD280
##  [70] NULL                 Hue                  Malic_Acid
##  [73] Ash_Alcanity         Malic_Acid           Ash
##  [76] Magnesium            Proline              Color_Intensity
##  [79] Magnesium            Proanthocyanins      NULL
##  [82] Ash                  Nonflavanoid_Phenols Malic_Acid
##  [85] Proanthocyanins      Hue                  Total_Phenols
##  [88] Malic_Acid           Flavanoids           NULL
##  [91] NULL                 Magnesium            Alcohol
##  [94] Ash_Alcanity         OD280                NULL
##  [97] Hue                  Magnesium            Proanthocyanins
## [100] Ash_Alcanity         Nonflavanoid_Phenols Proanthocyanins
## [103] Ash_Alcanity         Alcohol              Proline
## [106] Ash_Alcanity         Ash                  Magnesium
## [109] OD280                Proanthocyanins      Total_Phenols
## [112] Ash                  Ash_Alcanity         OD280
## [115] Proanthocyanins      Hue                  NULL
## [118] Ash_Alcanity         Proanthocyanins      Total_Phenols
## [121] Alcohol              NULL                 Malic_Acid
## [124] Nonflavanoid_Phenols Magnesium            Proanthocyanins
## [127] Color_Intensity      NULL                 Proline
## [130] NULL                 Magnesium            Nonflavanoid_Phenols
## [133] Ash_Alcanity         Ash                  Nonflavanoid_Phenols
## [136] NULL                 NULL                 Proline
## [139] Magnesium            NULL                 Nonflavanoid_Phenols
## [142] NULL                 Ash                  Color_Intensity
## [145] Hue                  NULL                 Proline
## [148] Magnesium            OD280                Ash
## [151] Proanthocyanins      Nonflavanoid_Phenols Alcohol
## [154] Proanthocyanins      Ash                  Ash_Alcanity
## [157] NULL                 Ash                  Proline
## [160] Color_Intensity      NULL                 NULL
## [163] Ash                  Color_Intensity      NULL
## [166] NULL                 Magnesium            Color_Intensity
## [169] Ash                  NULL                 NULL
## [172] NULL                 NULL                 Nonflavanoid_Phenols
## [175] Ash                  NULL                 Proanthocyanins
## Separation step:
##   [1]  0  0  0 10  9  0  8  9  7  8  6  7  5  0  0 10  9  8  7  0  0  0  8  6  7
##  [26]  4  6  8  9  7  0  5  6  3  6  0  5  4  0  0  7  6  5  0  8  7  0  0  9  0
##  [51]  0  0  8  6  2  6  8  7  5  8  7  6  7  4  9  8  9  7  6  0  5  6  3  5  7
##  [76]  6  4  6  5  7  0  8  6  9  8  9  7  8  1  0  0  6  5  6  4  0  6  5  7  6
## [101]  3  8  7  6  5  7  6  4  9  8  7  6  8  9  7  5  0  7  9  8  6  0  2  6  8
## [126]  7  5  0  7  0  6  7  4  7  8  0  0  9  6  0  8  0  7  5  9  0 10  8  7  8
## [151]  6  7  3  7  6  5  0  7  6  8  0  0  7  8  0  0  4  7  6  0  0  0  0  5  6
## [176]  0  7
##
## Available components:
## [1] "data"      "hasNA"     "order"     "variable"  "step"      "order.lab"
## [7] "call"
```

```
plot(mona_M)
```



In mona(), each division is based on a single (well-chosen) variable unlike other hierarchical methods like agnes & hclust. But the original dataset used is not in binary format, thus results of this model would be biased.

Final Thoughts on Hierarchical Clustering Algorithm For Hierarchical clustering, below are the two best models:

- The hierarchical model build using "single" linkage method and distance metric created by "manhattan" method is having low (around 0.548) clustering coefficient which suggest this model to be good for outlier detection. Thus I would suggests that this as a best model for outlier detection.(As this is based on a comparative scale)
- The hierarchical model build using "ward.D" clustering method and distance metric created by "canberra" method is having high (around 0.701) clustering coefficient which suggests that this model to be good for partitioning the data into equal sized groups.(As this is based on a comparative scale). Thus I consider this to be my best model for partitioning the data into equal sized groups.

Some Interesting observations

- Models built by agnes() and hclust() with average linkage show that row number 60 is distinct from all other observations suggesting it to be an outlier in the data. However since the dendograms of these models are not in the same height scale these are not chosen as final models.
- Model built using diana() has a clustering coefficient of 0.8 suggesting a good partition of data. By observing the dendogram created by diana() we can say that the data can be clustered into a equal sized group of 3-4.