Partitioning Cluster Analysis with Possibilistic C-Means

Zeynel Cebeci

2017-11-10

Contents

1 PREPARING FOR THE ANALYSIS

1.1 Install and load the package ppclust

This vignette is designed to be used with the '`ppclust`' package. You can download the recent version of the package from CRAN with the following command:

```
install.packages("ppclust")
```

If you have already installed '`ppclust`', you can load it into R working environment by using the following command:

```
library(ppclust)
```

1.2 Load the required packages

For visualization of the clustering results, some examples in this vignette use the functions from some cluster analysis packages such as '`cluster`', '`fclust`' and '`factoextra`'. Therefore, these packages should be loaded into R working environment with the following commands:

```
library(factoextra)
library(cluster)
library(fclust)
```

1.3 Load the data set

We demonstrate PCM on the Iris data set (Anderson, 1935) which is a real data set of the four features (Sepal.Length, Sepal.Width, Petal.Length and Petal.Width) of 150 iris flowers. The data set contains 50 samples from each of three iris species which is represented with the class variable in the last column of data set. One of these three natural clusters (Class 1) is linearly well-separated from the other two clusters, while Classes 2 and 3 have some overlap as seen in the plot below.

```
data(iris)
x=iris[,1:4]
head(x, 5)
```

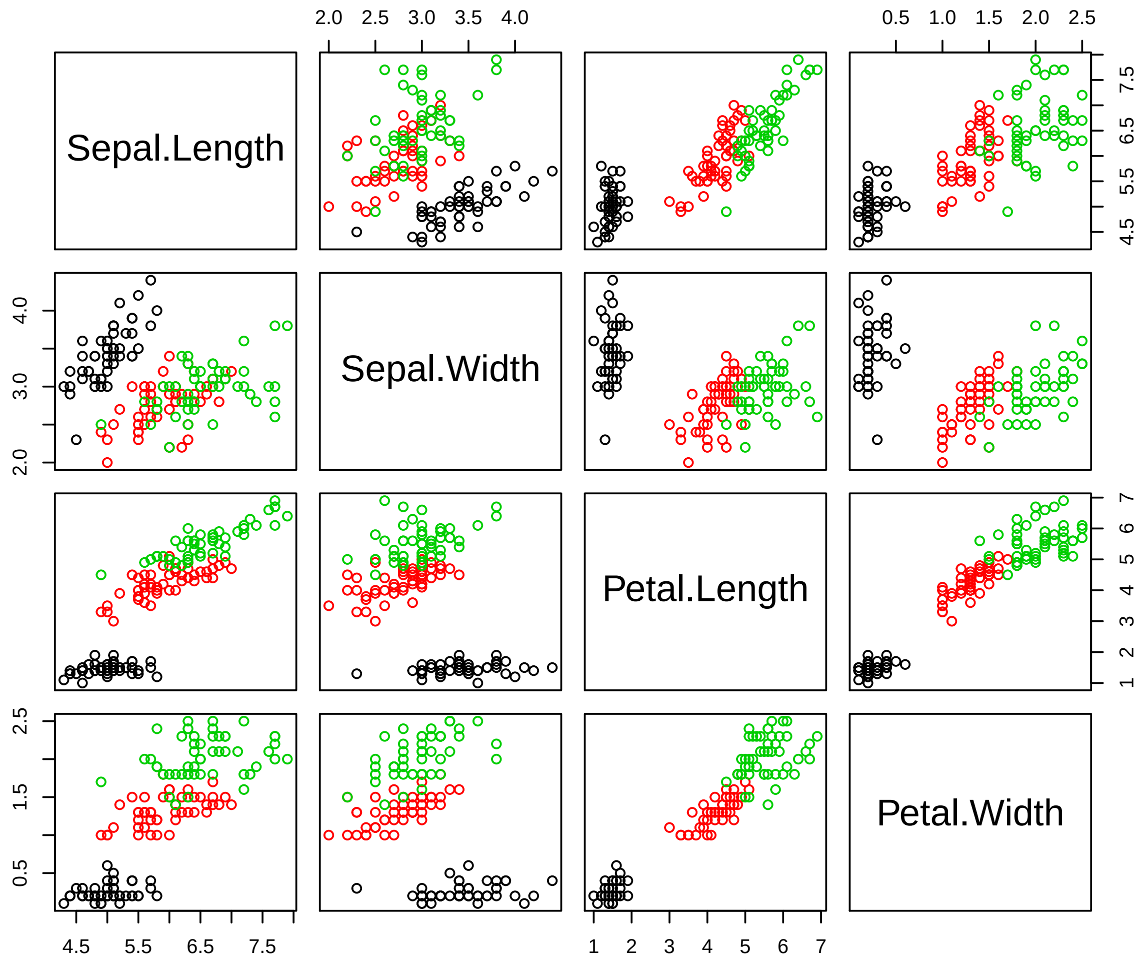
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

```
tail(x, 5)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
146	6.7	3.0	5.2	2.3
147	6.3	2.5	5.0	1.9
148	6.5	3.0	5.2	2.0
149	6.2	3.4	5.4	2.3
150	5.9	3.0	5.1	1.8

Plot the data by the classes of iris species

```
pairs(x, col=iris[,5])
```



2 POSSIBILISTIC C-MEANS CLUSTERING

Possibilistic C-Means (PCM) clustering algorithm was proposed by Krishnapuram and Keller (1993) in order to overcome the noise sensitivity problem of FCM. The algorithm differs from the other clustering algorithms that the resulting partition of the data can be interpreted as a possibilistic partition, and the membership values can be interpreted as degrees of possibility of the points belonging to the classes, i.e., the compatibilities of the points with the class prototypes (Krishnapuram & Keller, 1993). PCM relaxes the probabilistic constraint on the sum of the memberships of an object to all clusters in FCM. However, PCM must run on the fuzzy clustering results of FCM in order to calculate the vector $\vec{\Omega}$. Although it solves the noise sensitivity problem of FCM, the performance of PCM depends heavily on the initialization and often deteriorates due to the coincident clustering problem (Filippone et al, 2007).

2.1 Run PCM with Single Start

2.1.1 Initialization

In order to start PCM as well as the other alternating optimization algorithms, an initialization step is required to build the initial cluster prototypes matrix and fuzzy membership degrees matrix.

PCM is started with an integer specifying the number of clusters. In this case, the prototypes matrix is internally generated by using any of the prototype initalization techniques which are included in the package '`inaparc`'. The default initialization technique is *K-means++* with the current version of `pcm` function in this package. In the following code block, PCM runs for three clusters with the default values of the remaining arguments.

```r
res.pcm <- pcm(x, centers=3)
```

In order to generate the matrices of cluster prototypes and membership degrees, users can choose any of existing initialization algorithms in the R package '`inaparc`'. For this purpose, `alginitv` is used for assigning the cluster prototypes as shown for `inofrep` prototype initialization as follows:

```r
res.pcm <- pcm(x, centers=3, alginitv="inofrep")
```

Although the prototypes can be automatically initialized by using the function `pcm`, the users can also supply their own initial matrices as the input arguments of the function. In fact, PCM often suffers from stability problems if it is not initialized with terminal outputs of a corresponding probabilistic algorithm. Therefore, in order to compute some of the input arguments such as the cluster prototypes and mobilization scale parameter it needs fuzzy partitioning results of a probabilistic algorithm, i.e. FCM, (Timm et al, 2004).

```r
res.fcm <- fcm(x, centers=3)
res.pcm <- pcm(x, centers=res.fcm$v, memberships=res.fcm$u)
```

In the first line of above code chunk, FCM is run to obtain its terminal outputs and used as the input arguments of PCM as shown in the second line of the code chunk. If these arguments are not directly specified by the user they are automatically formed with an internal run of the FCM. But the function cancels to use the terminal outputs of FCM if the input argument `fcmrun` is set to `FALSE`.

In general, the squared Euclidean distance metric is used with PCM in order to compute the distances between the cluster centers and the data objects. Users can specify the other distance metrics with the argument `dmetric` as follows:

```r
res.pcm <- pcm(x, centers=3, dmetric="euclidean")
```

Although the argument eta (η), typicality exponent is usually choosen as 2 in most of the applications using PCM, users can increase η to higher values in order to make the typicalities more strict as follows:

```r
res.pcm <- pcm(x, centers=3, eta=4)
```

> There are other input arguments available with the function `pcm` to control the run of PCM algoritm. For the details, see `pcm` in the package manual of `ppclust`.

2.1.2 Clustering Results

2.1.2.1 Typicality Degrees Matrix

The typicality degrees matrix is the main output of the function `pcm`. The example code chuck below shows the top and bottom five rows of the typicality matrix.

```r
res.pcm <- pcm(x, centers=3)
head(res.pcm$t, 5)
```

```
##    Cluster 1 Cluster 2  Cluster 3
## 1 0.04442720 0.9212579 0.03779405
## 2 0.04415150 0.6576015 0.03755483
## 3 0.04070929 0.6720862 0.03460865
## 4 0.04352387 0.5748730 0.03701726
## 5 0.04342302 0.8722401 0.03693419
```

```
tail(res.pcm$t, 5)
```

```
##     Cluster 1  Cluster 2 Cluster 3
## 146 0.4170990 0.01601500 0.3766616
## 147 0.6962822 0.01912322 0.6586571
## 148 0.5961442 0.01746856 0.5550195
## 149 0.3731471 0.01607493 0.3346221
## 150 0.7421564 0.02041445 0.7084034
```

2.1.2.2 Initial and Final Cluster Prototypes

Initial and final cluster prototypes matrices can be achieved as follows:

```
res.pcm$v0
```

```
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Cluster 1     6.775011    3.052382     5.646782   2.0535467
## Cluster 2     5.003966    3.414089     1.482816   0.2535463
## Cluster 3     5.888932    2.761069     4.363952   1.3973150
```

```
res.pcm$v
```

```
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Cluster 1     6.172303    2.877981     4.763067   1.6077434
## Cluster 2     5.002622    3.398096     1.484792   0.2472753
## Cluster 3     6.172882    2.879010     4.763516   1.6065564
```

2.1.2.3 Summary of Clustering Results

The function `pcm` returns the clustering results as an intance of '`ppclust`' class. This object is summarized with the `summary` method of the package '`ppclust`' as follows:

```
summary(res.pcm)
```

```
## Summary for 'res.pcm'
##
## Number of data objects:  150
##
## Number of clusters:  3
##
## Crisp clustering vector:
##   [1] 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 2 2 2 2
##  [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 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
## [106] 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
## [141] 1 1 1 1 1 1 1 1 1 1
##
## Initial cluster prototypes:
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Cluster 1     6.775011    3.052382     5.646782   2.0535467
## Cluster 2     5.003966    3.414089     1.482816   0.2535463
## Cluster 3     5.888932    2.761069     4.363952   1.3973150
##
## Final cluster prototypes:
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Cluster 1     6.172303    2.877981     4.763067   1.6077434
## Cluster 2     5.002622    3.398096     1.484792   0.2472753
## Cluster 3     6.172882    2.879010     4.763516   1.6065564
##
## Distance between the final cluster prototypes
##              Cluster 1    Cluster 2
## Cluster 2 14.236634360
## Cluster 3  0.000003005 14.236634645
##
## Difference between the initial and final cluster prototypes
##           Sepal.Length Sepal.Width Petal.Length  Petal.Width
## Cluster 1 -0.602708313 -0.17440112 -0.883714980 -0.445803292
## Cluster 2 -0.001344113 -0.01599241  0.001976237 -0.006271055
## Cluster 3  0.283949533  0.11794095  0.399564041  0.209241338
##
## Root Mean Squared Deviations (RMSD): 0.7464666
## Mean Absolute Deviation (MAD): 4.190543
##
## Membership degrees matrix (top and bottom 5 rows):
##    Cluster 1 Cluster 2  Cluster 3
## 1 0.04442720 0.9212579 0.03779405
## 2 0.04415150 0.6576015 0.03755483
## 3 0.04070929 0.6720862 0.03460865
## 4 0.04352387 0.5748730 0.03701726
## 5 0.04342302 0.8722401 0.03693419
## ...
##     Cluster 1  Cluster 2 Cluster 3
## 146 0.4170990 0.01601500 0.3766616
## 147 0.6962822 0.01912322 0.6586571
## 148 0.5961442 0.01746856 0.5550195
## 149 0.3731471 0.01607493 0.3346221
## 150 0.7421564 0.02041446 0.7084034
##
## Descriptive statistics for the membership degrees by clusters
##           Size        Min        Q1      Mean    Median        Q3
## Cluster 1  100 0.08463109 0.2805857 0.4678317 0.4547212 0.6492502
## Cluster 2   50 0.18460456 0.4496620 0.6299153 0.6556992 0.7627964
## Cluster 3    0         NA        NA       NaN        NA        NA
##                 Max
## Cluster 1 0.9385043
## Cluster 2 0.9928247
## Cluster 3        NA
##
## Dunn's Fuzziness Coefficients:
## dunn_coeff normalized
##   0.490898   0.236347
##
## Within cluster sum of squares by cluster:
##       1       2
## 139.796  15.151
## (between_SS / total_SS =  75.39%)
##
## Available components:
##  [1] "t"          "v"          "v0"         "d"          "x"
##  [6] "cluster"    "csize"      "sumsqrs"    "k"          "eta"
## [11] "omega"      "iter"       "best.start" "func.val"   "comp.time"
## [16] "inpargs"    "algorithm"  "call"
```

All available components of the 'ppclust' object are listed at the end of summary. These elements can be accessed using as the attributes of the object. For example, the execution time of the run of PCM is accessed as follows:

```
res.pcm$comp.time
```

```
## [1] 0.55
```

2.2 Run PCM with Multiple Starts

In order to find an optimal solution, the function `pcm` can be started for multiple times. As seen in the following code chunk, the argument `nstart` is used for this purpose.

```
res.pcm <- pcm(x, centers=3, nstart=5)
```

When the multiple start is performed, either initial cluster prototypes or initial membership degrees could be wanted to keep unchanged between the starts of algorithm. In this case, the arguments `fixcent` and `fixmemb` are used for fixing the initial cluster prototypes matrix and initial membership degrees matrix, respectively.

```
res.pcm <- pcm(x, centers=3, nstart=5, fixmemb=TRUE)
```

Both the prototypes and memberships cannot be fixed at the same time.

2.2.1 Display the best solution

The clustering result contains some outputs providing information about some components such as the objective function values, number of iterations and computing time obtained with each start of the algorithm. The following code chunk demonstrates these outputs.

```
res.pcm$func.val
```

```
## [1]  170.2677  904.5607  904.5607 1176.7768  492.8166
```

```
res.pcm$iter
```

```
## [1] 74 73 72 38 38
```

```
res.pcm$comp.time
```

```
## [1] 0.57 1.04 1.06 0.78 0.75
```

Among the outputs from succesive starts of the algorithm, the best solution is obtained from the start giving the minimum value of the objective function, and stored as the final clustering result of the multiple starts of PCM.

```
res.pcm$best.start
```

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

2.2.2 Display the summary of clustering results

As described for the single start of PCM above, the result of multiple starts of PCM is displayed by using `summary` method of the 'ppclust' package.

```
summary(res.pcm)
```

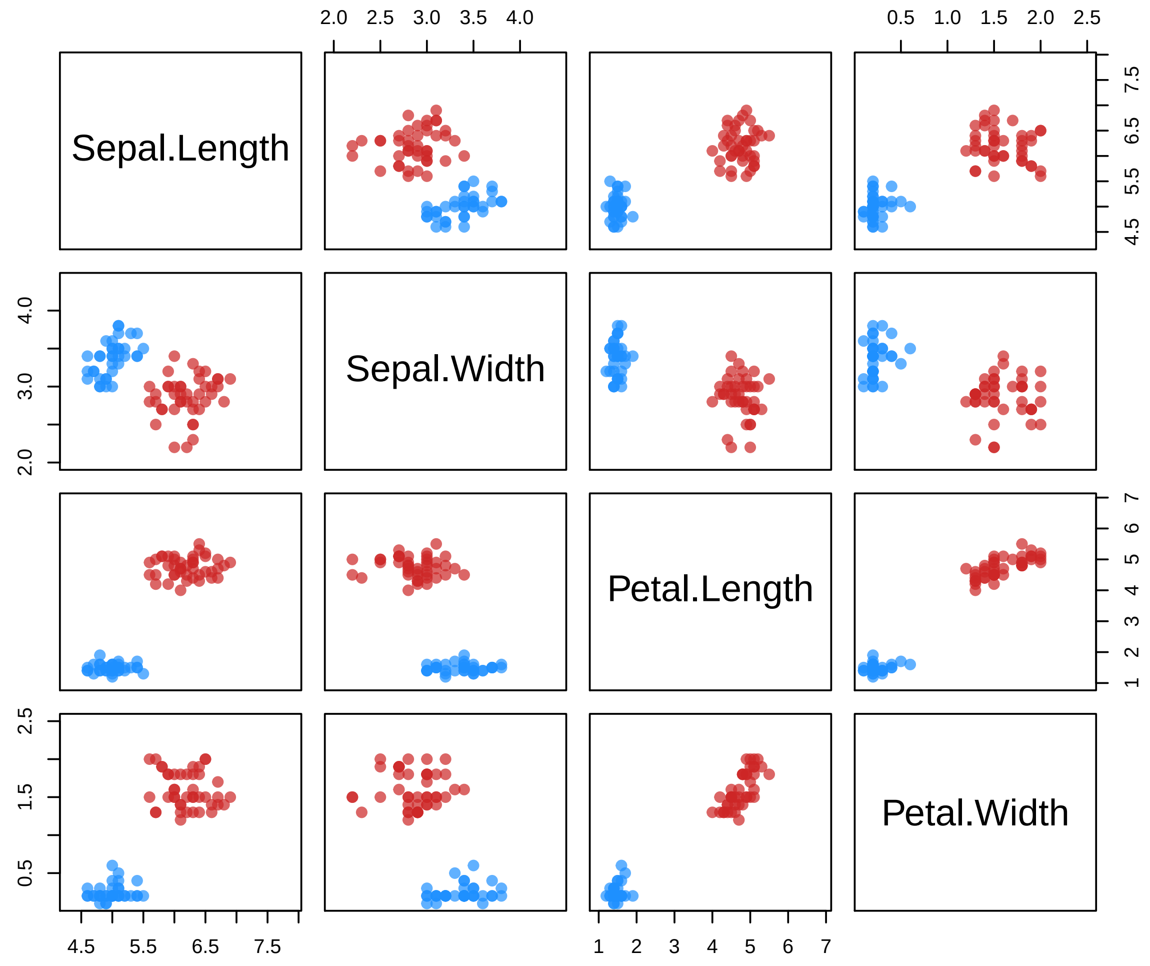
```
## Summary for 'res.pcm'
##
## Number of data objects:  150
##
## Number of clusters:  3
##
## Crisp clustering vector:
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 2 2 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 2 2 2 2
## [106] 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 2 2 2 2
## [141] 2 2 2 2 2 2 2 2 2 2
##
## Initial cluster prototypes:
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Cluster 1          5.4         3.0          4.5         1.5
## Cluster 2          6.9         3.1          4.9         1.5
## Cluster 3          4.6         3.6          1.0         0.2
##
## Final cluster prototypes:
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Cluster 1     5.002622    3.398096     1.484792   0.2472753
## Cluster 2     6.172303    2.877981     4.763067   1.6077434
## Cluster 3     6.172882    2.879010     4.763516   1.6065564
##
## Distance between the final cluster prototypes
##              Cluster 1    Cluster 2
## Cluster 2 14.236634360
## Cluster 3 14.236634645  0.000003005
##
## Difference between the initial and final cluster prototypes
##           Sepal.Length Sepal.Width Petal.Length Petal.Width
## Cluster 1   -0.3973782   0.3980964   -3.0152082  -1.2527247
## Cluster 2   -0.7276971  -0.2220188   -0.1369332   0.1077434
## Cluster 3    1.5728819  -0.7209897    3.7635157   1.4065564
##
## Root Mean Squared Deviations (RMSD): 3.200136
## Mean Absolute Deviation (MAD): 18.29566
##
## Membership degrees matrix (top and bottom 5 rows):
##   Cluster 1  Cluster 2  Cluster 3
## 1 0.9212579 0.04442720 0.03779405
## 2 0.6576015 0.04415150 0.03755483
## 3 0.6720862 0.04070929 0.03460865
## 4 0.5748730 0.04352387 0.03701726
## 5 0.8722401 0.04342302 0.03693419
## ...
##      Cluster 1 Cluster 2 Cluster 3
## 146 0.01601500 0.4170990 0.3766616
## 147 0.01912322 0.6962822 0.6586571
## 148 0.01746856 0.5961442 0.5550195
## 149 0.01607493 0.3731471 0.3346221
## 150 0.02041446 0.7421564 0.7084034
##
## Descriptive statistics for the membership degrees by clusters
##           Size        Min        Q1      Mean    Median        Q3
## Cluster 1   50 0.18460456 0.4496620 0.6299153 0.6556992 0.7627964
## Cluster 2  100 0.08463109 0.2805857 0.4678317 0.4547212 0.6492502
## Cluster 3    0         NA        NA       NaN        NA        NA
##                 Max
## Cluster 1 0.9928247
## Cluster 2 0.9385043
## Cluster 3        NA
##
## Dunn's Fuzziness Coefficients:
## dunn_coeff normalized
##   0.490898   0.236347
##
## Within cluster sum of squares by cluster:
##       1       2
##  15.151 139.796
## (between_SS / total_SS =  75.39%)
##
## Available components:
##  [1] "t"          "v"          "v0"         "d"          "x"
##  [6] "cluster"    "csize"      "sumsqrs"    "k"          "eta"
## [11] "omega"      "iter"       "best.start" "func.val"   "comp.time"
## [16] "inpargs"    "algorithm"  "call"
```

3 VISUALIZATION OF THE CLUSTERING RESULTS

3.1 Pairwise Scatter Plots

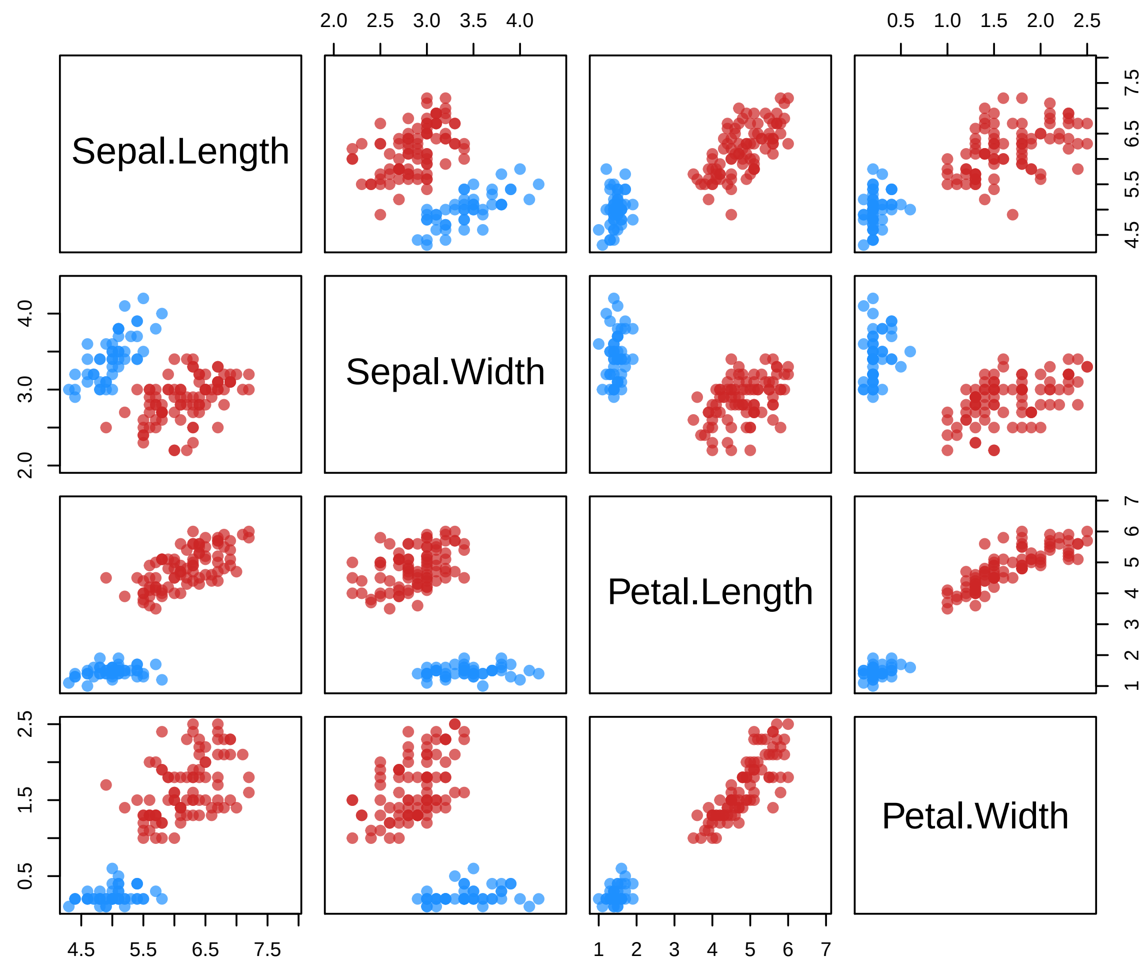
There are many ways of visual presentations of the clustering results. One common techique is to display the clustering results by using scatter plots of pairs of the features. For this purpose, the `plotcluster` can be used to plot the clustering results with the color palette 1 and transparent colors as follows:

```
plotcluster(res.pcm, cp=1, trans=TRUE)
```



In the above figure, the plot shows the objects which have typicality degrees greater than 0.5. In order to filter the objects for different typicality degrees, the value of `tv` argument can be changed as follows:

```
plotcluster(res.pcm, cp=1, tv=0.2, trans=TRUE)
```

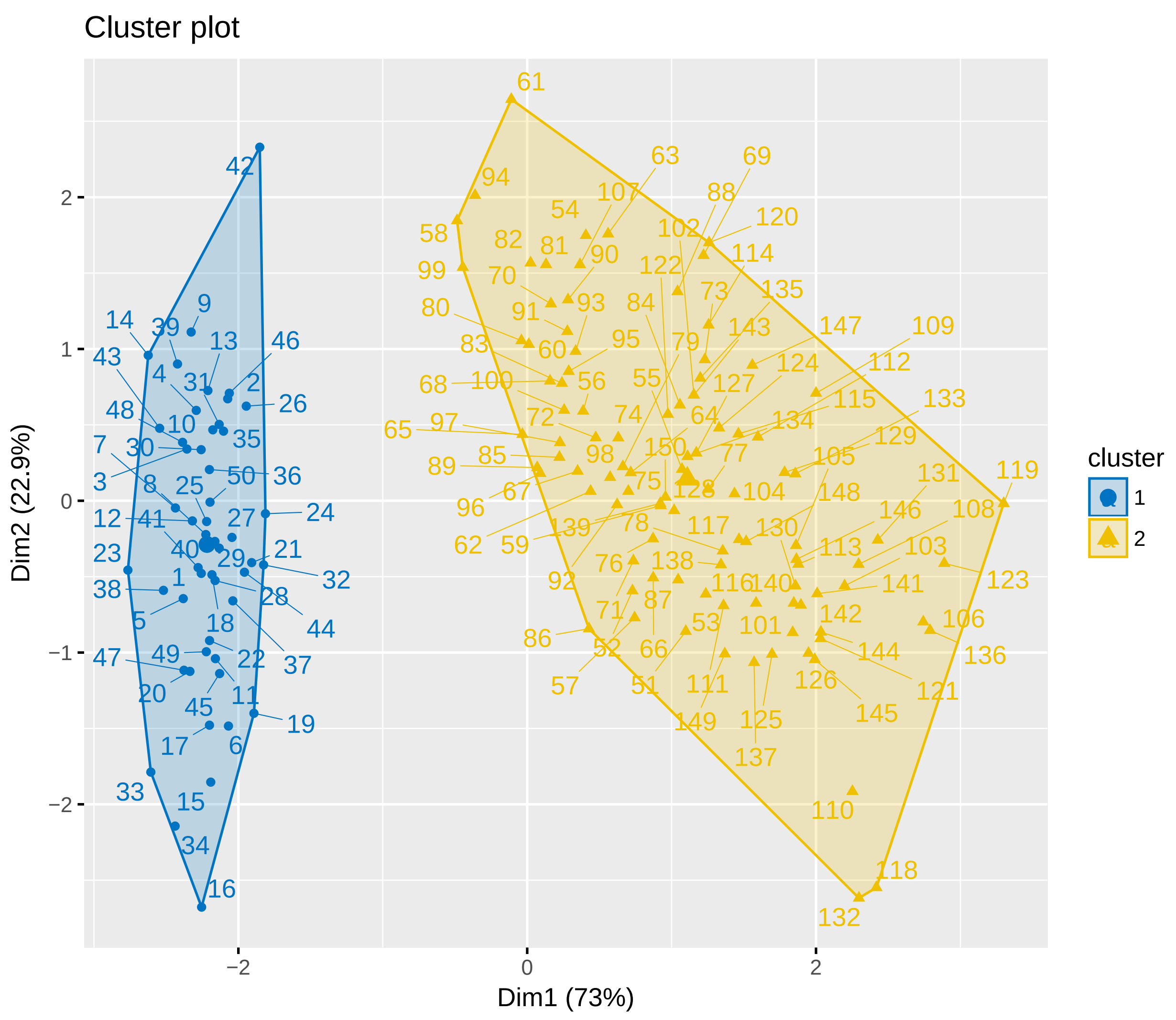


The details can be viewed on the package manual for the function `plotcluster`.

3.2 Cluster Plot with fviz_cluster

There are some nice versions of the cluster plots in some of the R packages. One of them is the function `fviz_cluster` of the package '`factoextra`' (Kassambara & Mundt, 2017). In order to use this function for the fuzzy clustering result, first the fuzzy clustering object of `ppclust` should be converted to `kmeans` object by using the `ppclust2` of the package '`ppclust`' as shown in the first line of the code chunk below:

```
res.pcm2 <- ppclust2(res.pcm, "kmeans")
factoextra::fviz_cluster(res.pcm2, data = x,
  ellipse.type = "convex",
  palette = "jco",
  repel = TRUE)
```

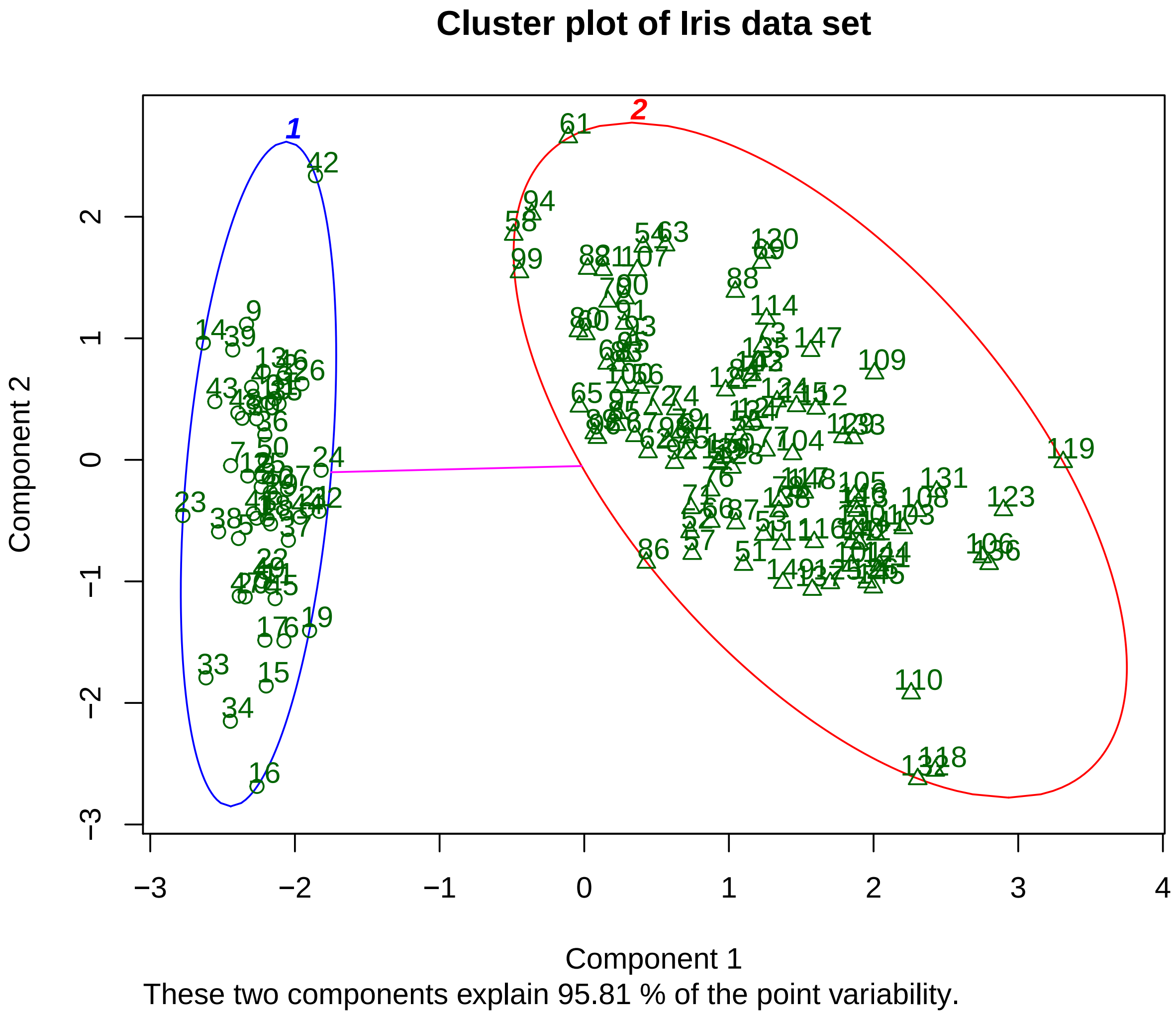


3.3 Cluster Plot with clusplot

User can also use the function `clusplot` in the package '`cluster`' (Maechler et al, 2017) for plotting the clustering results. For this purpose, the fuzzy clustering object of `ppclust` should be converted to `fanny` object by using the `ppclust2` function of the package '`ppclust`' as seen in the following code chunk.

```
res.pcm3 <- ppclust2(res.pcm, "fanny")

cluster::clusplot(scale(x), res.pcm3$cluster,
  main = "Cluster plot of Iris data set",
  color=TRUE, labels = 2, lines = 2, cex=1)
```



References

Anderson, E. (1935). The irises of the GASPE peninsula, in *Bull. Amer. Iris Soc.*, 59: 2-5.

Bezdek J.C. (1981). *Pattern recognition with fuzzy objective function algorithms.* Plenum, NY. https://isbnsearch.org/isbn/0306406713

Krishnapuram, R. & Keller, J.M. (1993). A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1(2):98-110.

Krishnapuram, R. & Keller, J.M. (1996). The possibilistic c-means algorithm: insights and recommendations. *IEEE Transactions on Fuzzy Systems*, 4(3):385-393.

Timm, H., Borgelt, C., Döring, C. & Kruse, R. (2004). An extension to possibilistic fuzzy cluster analysis. *Fuzzy Sets and Systems*, 147 (1): 3-16.

Filippone, M., Masulli, F., & Rovetta, S. (2007). Possibilistic clustering in feature space. In *Int. Workshop on Fuzzy Logic and Applications*, pp. 219-226. Springer, Berlin, Heidelberg.

Ferraro, M.B. & Giordani, P. (2015). A toolbox for fuzzy clustering using the R programming language. *Fuzzy Sets and Systems*, 279, 1-16. http://dx.doi.org/10.1016/j.fss.2015.05.001

Kassambara, A. & Mundt, F. (2017). factoextra: Extract and Visualize the Results of Multivariate Data Analyses. R package version 1.0.5. https://CRAN.R-project.org/package=factoextra

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M. & Hornik, K.(2017). cluster: Cluster Analysis Basics and Extensions. R package version 2.0.6. https://CRAN.R-project.org/package=cluster