**1 Introduction**

In this project, a novel framework is implemented with R language to elucidate core classes.

The framework first pre –processing the dataset and then apply 3 different clustering algorithm (Kmeans, FCM, PAM) with various numbers of centers. The results are compared to get the best number of clusters. Then with that cluster number, the labels of each instance assigned by different clustering algorithm are obtained. Focusing on the cluster correspondences, core classes containing the biggest possible number of data points were defined. Only in the condition that all the clustering algorithms assign the same type of label, will the instance be allocated to that label. The instance will be assigned with “NC” if any contradiction occurs. In the end, the biplot of the final clustering result will be represented.

**2 Implementation**

**Pre processing**

Pre processing :matrix, numerical, blank replacement

In order to let the all data which was passed in by user can be manipulated by clustIndex function, we need to do few pre-processing in advance.

1. Make the data be matrix

dataset <- as.matrix(x[,ccol])

According to the column vector that has been passed in, program will pick up the data that have

been chosen by user and transform it into matrix data structure.

2. Delete NA data

if(length(dataset) > length(na.omit(dataset)))

{

dataset <- na.omit(dataset)

warning("There are some missing data in your choosing range, they have been removed

from matrix")

}

Because NA data will be automatically removed by na.omit() function, so the program will

give user warning for NA deletion.

3. Ensure the numerical data

for(i in length(isNum))

{

if(isNum[i] == FALSE)

stop("Sorry, the data you chosen must be numeric.")

}

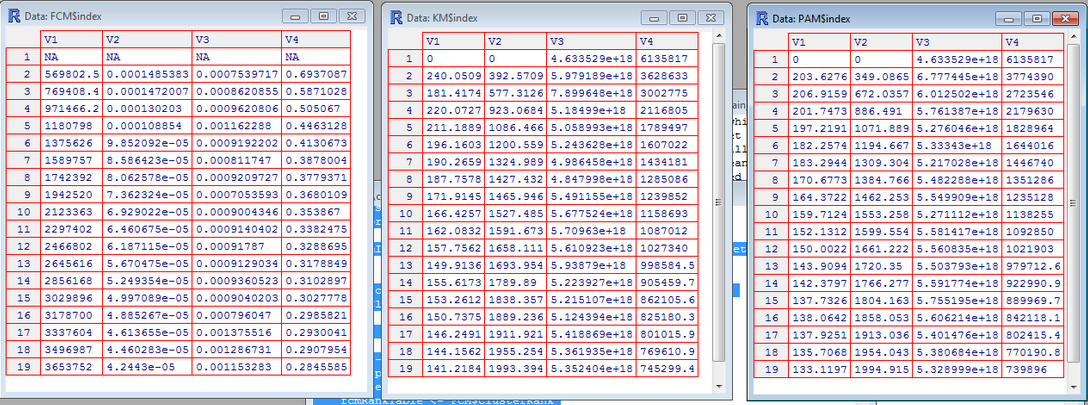
**Cluster number choosing**

Input: dataset need to processed, maximum number of clusters

Output: the best number of clusters

We use three different algorithms, namely as K-means, PAM, and Fuzzy C-means, to cluster the data. In order to choose the best amount of clusters, our program will follow the following process:

1. We use clustIndex function for PAM and K-mans and fclustIndex function for Fuzzy C-means to generate validity index tables which can evaluate the quality of different number of clusters. And program will automatically generate three tables for each algorithm.
2. Using addClusterOne function add default index to the first row of index table which used to be NA for one cluster.
3. Pick up Calinski and Harabasz index, Scott and Symons index, Marriot index, TraceW index and TraceW-1B index from PAM index table and K-means index table. And pick up Fuzzy Hypervolume index, Partition Density index, Xie-Beni index and Partition Coefficient index from FCM index table.
4. Rank the table using difference between adjacent cluster numbers’ index.
5. Using bestCluster function which uses original weighted average algorithm to output the best number of cluster.



Detail

1. addClusterOne function

PAM and K-means use the same type of index. This table includes 15 different indexes. In usual cases, the best indexes we choose to evaluate PAM and K-means are following six indexes:

(SSB: Total sum of squares between, SSW: Total sum of squares within)

* Calinski-Harabasz index:
  + (SSB/(k-1))/(SSW/(n-k))
  + where n is the number of data points and k is the number of clusters.
* hartigan index:
  + log(SSB/SSW)
* scott and Symons index:
  + n\*log(|T|/|W|)
  + where n is the number of data points and |\cdot| stands for the determinant of a matrix.
* marriot index:
  + k^2 |W|
  + where k is the number of clusters.
* tracew index:
  + Trace W
* friedman index:
  + Trace W^{(-1)} B
  + where B is the scatter matrix of the cluster centers.

Because we are using either the maximum or the minimum decision rule value, when we calculate the maximum or the minimum index value of two clusters, we need the index value of one cluster. However, among all fifteen indexes, one cluster’s index value is meaningless both in reality and mathematics, such as Hartigan index. When n equals to one, Hartigan index will approach negative infinity which is meaningless in numerical calculation. So we decided to ignore Hartigan index, only use five indexes for comparison in results of K-means and PAM.

1. bestCluster function

This function use original algorithm which based on weighted distribution of different indexes of different algorithm. The weight in each algorithm is same, and the final score will based on the following equation:

Here means the when data divided into i clusters’ score.

means the data of K-means’ rank table’s k row and i-1 row.

means the data of PAM’s rank table’s k row and i-1 row.

means the data of Fuzzy C-means’ rank table’s k row and i-1 row.

Here I will mainly introduce how I calculate and .

We have six data sets which have already been given the best number of cluster:

|  |  |
| --- | --- |
|  | Best number of cluster |
| magdy\_first\_grade\_surv\_no\_sirt1 | 3 |
| alaa\_3\_markers | 3 |
| alaa\_4\_markers | 3 |
| binafsha | 6 |
| caroline\_and\_stuart | 3 |
| sahar\_g3 | 3 |

Because K-means and PAM algorithm use the same index, so I chose they use the same parameter , and the Fuzzy C-means rank’s parameter will be (1-2 \*)

And we run from 0.25 to 0.5 with 0.001 as interval. The best number of cluster has the minimum value of this cluster.

Compare the best number of cluster using different parameter with best number of clusters table.

Final parameters we run are approximately: = 3/8, = 1/4

**Apply clustering algorithm**

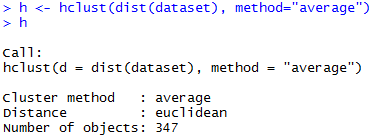
After the best number of cluster is known, we apply that number to the three clustering algorithms.

The hierarchical tree is also created for obtaining the centers under various numbers of clusters, which will be used in kmeans and FCM. These functions can be run with fix number of centers as the initial points. The hierarchical tree can divide the data into different levels with the best centers.

number = cluster\_num(dataset, cluster)

#create the tree of the orginal tree

h <- hclust(dist(dataset), method="average")



The hierarchical tree is applied to kmeans and FCM in order to get the centers under various numbers of clustesr.

##Kmeans

initial <- tapply(dataset, list(rep(cutree(h,number), ncol(dataset)), col(dataset)), mean)

cl\_km <- kmeans(dataset,initial, iter)

km <- data.frame(dataset, clust=cl\_km$cluster)

##pam

cl\_pam <- pam(dataset, number, diss = inherits(dataset, "dist"), metric = dist,

medoids = NULL, stand = FALSE, cluster.only = FALSE)

pam <- data.frame(dataset, clust=cl\_pam$clustering)

##FCM

initial <- tapply(dataset, list(rep(cutree(h,number), ncol(dataset)), col(dataset)), mean)

cl <- cmeans(dataset,initial,iter,verbose=FALSE,dist= dist, method="cmeans",m=2,initial)

fcm <- data.frame(dataset, clust=cl$cluster)

The three labels created by three different clustering algorithms are concatenated with the original dataset.

#construct all results together

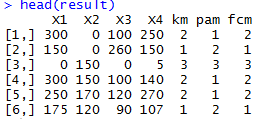
classif <- data.frame(cbind(km$clust, pam$clust, fcm$clust))

names(classif) <- c("km", "pam", "fcm")

result <- data.frame(dataset,classif)

result = as.matrix(result)

figure xx shows the new table which contains the labels given by different clustering algorithms

****

**Align Label**

The cluster labels need to be aligned in order to have the same patient assigned to the cluster named in the same way by different algorithms. However, in most of cases, the two similar groups are assigned with different labels. One way to solve that problem is to compare the distance of each cluster centers and adjust previous labels.

The centers of three different groups from three clustering algorithms are calculated. We create 3 dimension arrays to store the result.

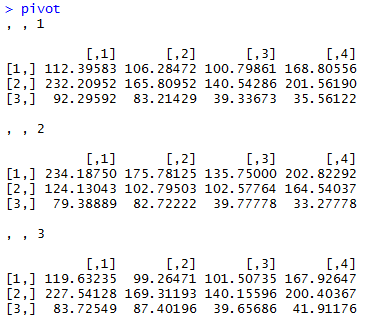
for(w in 1:length\_label){

for(u in 1:l){

pivot[u,,w] <-colMeans(pick(dataset,label[w],u,attr))

}

}



The centers of the clusters made by kmeans are used as the reference, because kmeans is more widely used.

Then we calculate the centers of other labels and compute the distances matrix. The result is ranked and stored into a new array.

#initialize the rank array

rank = array(0,c(l,length\_label-1))

#create the table of the rank of the distance to referential centers

for(a in 2:length\_label){

for(b in 1:l){

min = Inf

index = 0

for(c in 1:l){

distant = dis(pivot[c,,1],pivot[b,,a])

if(distant < min){

min = distant

index = c

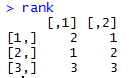
}

}

rank[b,a-1] = index

}

}



Previous labels are modified based on the ranking table above. Finally we get a table whose labels are aligned.

#according to the rank table, assign the new value to the original labels.

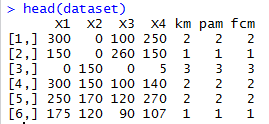
for(ccc in 1:length) {

for(aaa in 1:(length\_label-1)){

dataset[ccc,label[1]+aaa] = rank[dataset[ccc,label[1]+aaa],aaa]

}

}



**Define core class**

An empty vector is created to store the final clustering result. The agreement of three clustering results is checked one by one. If any label is different with the other two , that instance’s label will be rewrite with “NC”. Otherwise, the original label does not change.

interss = c()

for(i in 1:length){

if(aligned\_result[i,wide+1] == aligned\_result[i,wide+2] & aligned\_result[i,wide+2] == aligned\_result[i,wide+3] ){

interss = c(interss,as.integer(aligned\_result[i,wide+1]))

}else{

interss = c(interss,"NC")

}

}



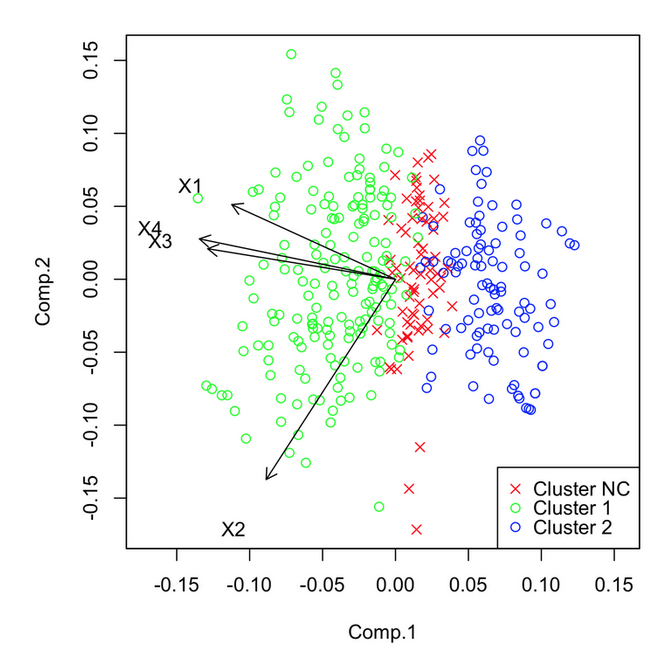


**Implementation of plotting**

Input: dataset which has already been clustered, best number of clusters

Output: The plot which contains all data points

In this part, program mainly using bip function which includes bips function and rely on the ouput of cluster\_color function to draw different shapes’ and colours’ points on the coordinate axis to present different clusters’ data.



**Warnings & Stops**

1. warning(“There are some missing data ins your choosing range, they have been removed from matrix ”)
2. stop(“Sorry, the data you chosen must be numeric”)

Explanation:

In order to generate the index table for different number of clusters, the dataset we are passed in should satisfy following condition:

1. Dataset user passed in have no NA data.
2. All data should be numeric.

If dataset contains NA data, it will be automatically removed from initial dataset, and deliver a warning to users.

If data contains none numeric data, the program will crush on the certain step in processing. It will directly stop the program and deliver a warning.

1. warning("’scale’ is outside [0,1]”) line 60
2. warning(paste(“The cluster of current dataset cannot be set more than”, cluster))

Explanation:

Not every dataset user passed in can be divided properly to the limitation user gave. For example, user passed in 19 as cluster limitation, program finds that this data can only be divided in to 15 clusters. When there is 15 clusters, there is one cluster only including one datum. So index of that condition cannot be calculated. In this situation, user will be informed, and program will automatically replace the cluster number limitation with the new one.

1. if("try-error" %in% class(try(suppressWarnings(clustIndex(cl,dataset,index="all")))))

{

cluster <- i - 2

if(cluster < 2)

{

stop("The indexes cannot be calculated")

}

isCluster <- TRUE

warning(paste("The cluster of current dataset cannot be set more than",cluster))

break

}

Explanation:

Not every dataset can be divided properly to calculate the indexes for the certain number of clusters. When we apply the breast cancer data inside, the indexes of K-means algorithm can’t be calculated more than 2 clusters. It will give error message:

Error in solve.default(zttw$w): Lapack routine dgesv: system is exactly singular: U[6,6] = 0

The error message is telling you that your matrix is singular and cannot be inverted. But some datasets cause this error still can be applied for cclass package, only if the number of clusters is more than 4. Because our function is based on differences between indexes of different number of clusters. If we have the indexes of 1, 2 and 3 clusters, it will still can give us the score of 2 clusters. So though the error will be thrown, it is still running.

**3 Package Constructions**

Finally the main function is transformed into a package. Follow the standard procedure, a generic function is written, which contains only a call to UseMethod.

#abstract function

cclass <- function(x, ...) UseMethod("cclass")

Then our code has been is put into the default function of cclass.

#realize the abstract function

cclass.default = function (x, ccol = c(1:dim(x)[2]),cluster=19, iter = 100, dist="euclidean")

Finally the print function is written to make the output look more beautiful, which disappears the best number of clusters, and the components of each clustering result.

print.cclass <- function(x)

{

wide = dim(x$cluster)[2]

table = table((x$cluster)[,wide])

length\_label = length(table)

cat("The best number of clusters is ", x$number,"\n\n")

for(i in 1:(length\_label-1) ){

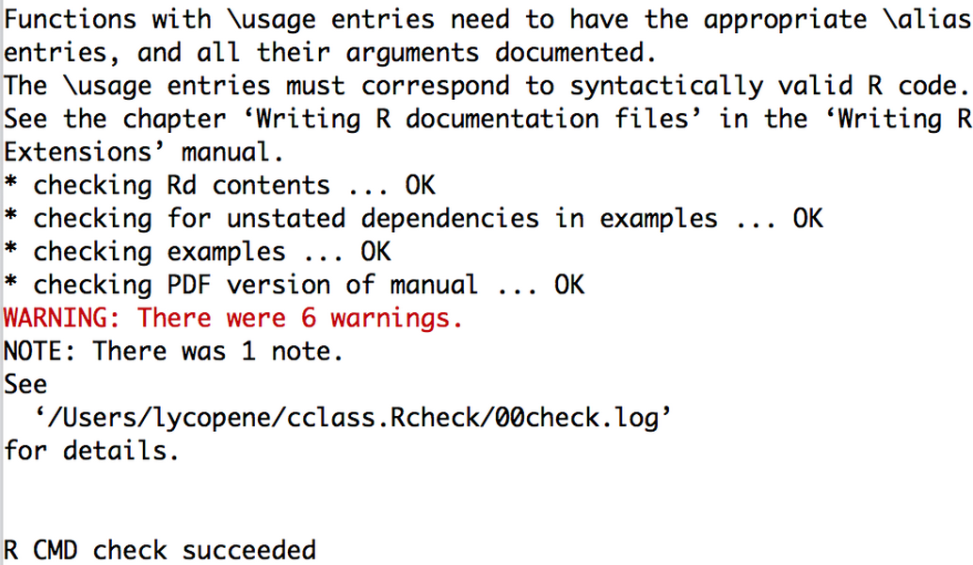
cat("The number of instances assigned to label", i,"is",table[i],"\n")

}

cat("The number of instances assigned to NC is",table[length\_label],"\n")

}

Finally we clean our session and start creating the package. We first use the command prompt() to create the file contains source code and DESCRIPTION document. Then we implement R package tool to check the validity of the file with command “R CMD check”.

****

After we make sure everything is correct, we start building the package using command “R CMD INSTALL”.

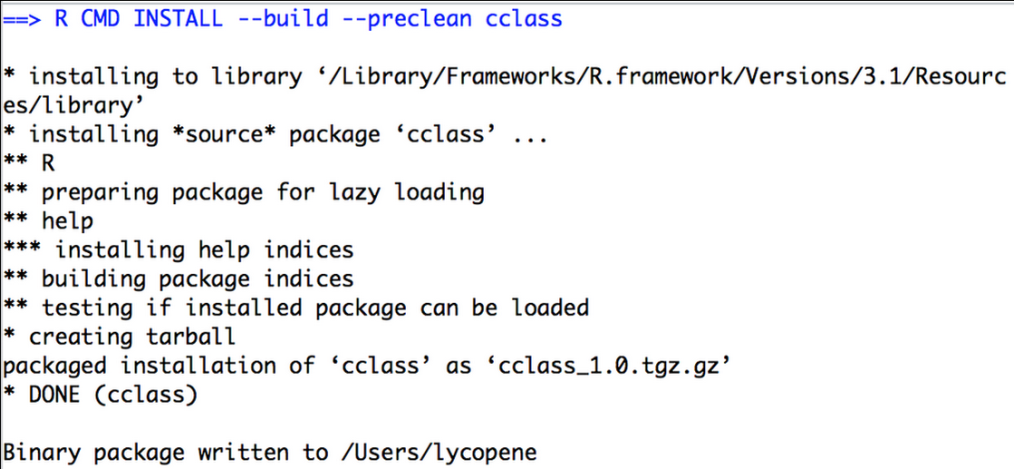
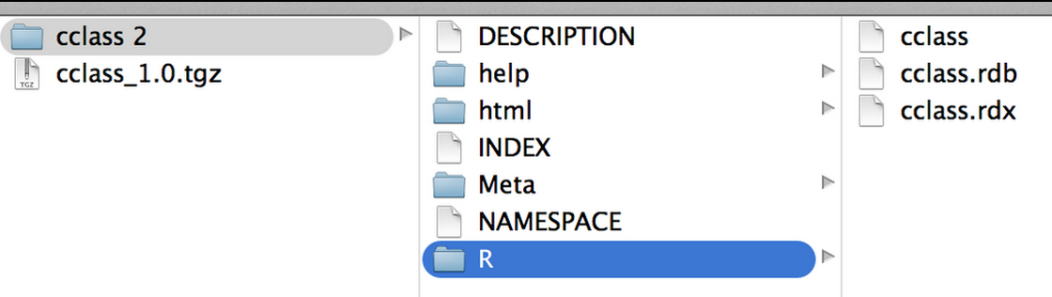
****

Figure xx shows the package we created “cclass\_1.0.tgz” and its uncompressed file.

****

**Future work**

More clustering algorithm

As we introduced before, right now we using HCA algorithm help us to select centre points for each cluster, then use K-means, PAM and Fuzzy C-means to cluster the dataset.

This method help users pre-process the data. The precise degree is depends on the number of the algorithm we use. But when we were doing research, it is not hard to find that different algorithm using different principles. So if too many algorithm are used, it may cause some side-effects. Therefore, the precise level of the core class framework is not completely in direct proportion to number of algorithm we use.

Thus, in the future, in order to increase the precise degree of our framework, researchers can focus on how to pick some right algorithm which may depend on identity of dataset.

Index for one cluster

For we are using the either the maximum or the minimum decision rule value for the difference between indexes of different number of clusters, when we calculate the score for two clusters, we need the index for one cluster. But in some cases, one cluster’s index is not meaningful both in numeric and mathematics.

In the future, researcher can focus on choosing some other index which has meaningful index for one cluster.

Choose best cluster number

Right now, the score is calculated by the weighted average of ranks of different indexes. Though rank help us simplify the process of quantize the different indexes. However, it still cannot represent the number behind properly. For the following reasons:

1. No matter how big difference between neighbour index, it only gets 1 as their difference.
2. Different indexes is weighted differently. Only categorized parameter by different algorithm is far away from precise number.

**Align function**

In the section of label alignment, the referential label we choose as the standard is kmeans. However, we are not sure whether, as the number of labels and clustering algorithms increasing, whether it will lead to different result when different referential label is selected. What we can do is to add another big loop outside our previous code and run it by using different clustering algorithm as reference. The performances under different referential label may various.

**4 Appendixes**

Package: cclass  
Type: Package  
Title: Core class selection framework  
Version: 1.0  
Date: 2014-08-02  
Author: Xiaotian Zhou, Chunfan Gao, Daniele.Soria  
Maintainer: Xiaotian Zhou<xiaotian.zhou@oriel.ox.ac.uk>  
Description: This package contain a framework to elucidate a set of core groups of a dataset, which specifies the application of different clustering  
algorithms, serval statistical and visualisation techniques. At the beginning of this framework, different clustering algorithms are applied. Then a  
set of common classes is defined in order to determine the fundamental characteristics of data expressed by different groups.  
Suggests: cclust, e1071, cluster  
License: GPL-2

**5 References**

P. Kellam, X. Liu, N. Martin, C. Orengo, S. Swift, and A. Tucker, “Comparing, contrasting and combining clusters in viral gene expression data,” inProceedings of 6th Workshop on Intelligent Data Analysis in Medicine, 2001.

Soria, D. Garibaldi, J.M. "A novel framework to elucidate core classes in a dataset" Evolutionary Computation (CEC), 2010 IEEE Congress on. Date of Conference: 18-23 July 2010 Page(s): 1 - 8 Print ISBN: 978-1-4244-6909-3 INSPEC Accession Number: 11567584 Conference Location : Barcelona DOI: 10.1109/CEC.2010.5586331 Publisher: IEEE

Friedrich Leisch “Creating R Packages: A Tutorial” http://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf