User Guide for CLARA.seq

Louis Tochon

06/11/2021

Contents

Introduction

CLARA.seq is a R-package to cluster big datasets of sequences. It uses the TraMineR (http://cran.r-project.org/web/packages/TraMineR/) package to calculate distances between each sequence but this package is made for 32-bits computers, which implies that the quantity of objects is limited to 46'300 objects (because we need to compute a matrix of size $n * n$). CLARA.seq provides three different functions : CLARA, CLARANS and ROBUST FUZZY C-MEDOIDS.
The package has an index called Davies-Bouldin index to check the validity of the clustering found after the use of a clustering method.

Use of the functions

CLARA

CLARA has been implemented with a parameter which can determine which aspect of the clustering is the most important. Indeed the parameter *find_best_method* allows users to choose between :

- **Distance** : finding the best clustering with the mean distance method which aims to minimize the distance between each object and his medoid
- **DB** : finding the best clustering with the Davies-Bouldin index method which aims to minimize the index of the global clustering

Research of the best clutering with the mean distance method

Here is an example of the call of this function :

```
my_clustering <- clara_clust(seq[1:100,],nb_cluster = 4, nb_sample = 20, size_sample = 20
+   find_best_method = "Distance")
```

Research of the best clutering with the Davies-Bouldin index method

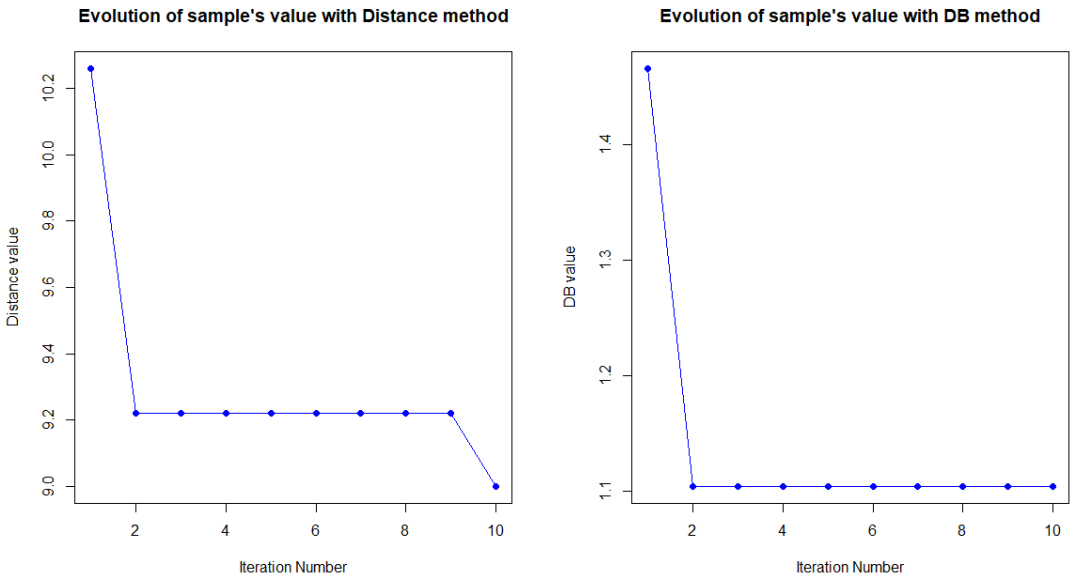
Here is an example of the call of this function :

```
my_clustering <- clara_clust(seq[1:100,],nb_cluster = 4, nb_sample = 20, size_sample = 20
+   find_best_method = "DB")
```

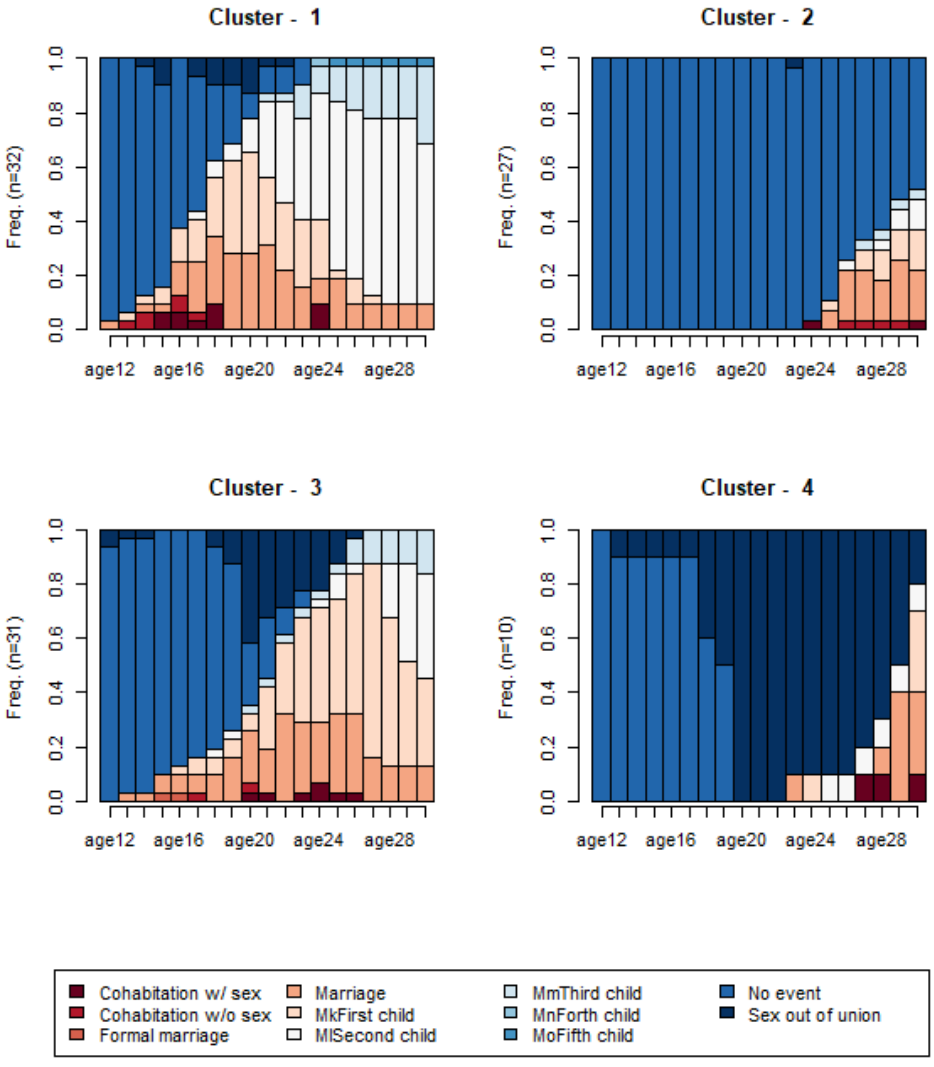
Outputs

Plots This function provide two types of graph as outputs. These graphs can be enabled through the *plot* parameter.

- The first graph is the representation of the value evolution for each iteration (depending of the method used) :



- The second one shows the final clustering of the dataset :



Console outputs The function print for the users the details of the run and the calculation time needed :

```
CLARA ALGORITHM Improved


Table of Sample Values with DB Method

N°1  (Med: 126, 20, 65, 194)   : 1.5755  |  N°2  (Med: 91, 88, 182, 66)   : 1.1019
N°3  (Med: 182, 61, 138, 201)  : 1.4963  |  N°4  (Med: 115, 171, 183, 1)  : 1.4954
N°5  (Med: 117, 26, 127, 101)  : 1.4565  |  N°6  (Med: 75, 31, 106, 110)  : 1.7497
N°7  (Med: 149, 137, 163, 179) : 1.3346  |  N°8  (Med: 22, 117, 7, 66)    : 1.2015
N°9  (Med: 171, 155, 54, 68)   : 1.2720  |  N°10 (Med: 139, 77, 183, 155) : 1.8443
N°11 (Med: 155, 77, 74, 146)   : 1.4616  |  N°12 (Med: 26, 10, 174, 75)   : 1.6014
N°13 (Med: 93, 52, 115, 105)   : 1.4082  |  N°14 (Med: 139, 194, 36, 116) : 1.4681
N°15 (Med: 149, 51, 90, 19)    : 1.3044  |  N°16 (Med: 19, 115, 88, 121)  : 1.4710
N°17 (Med: 77, 146, 71, 96)    : 1.4134  |  N°18 (Med: 95, 91, 171, 126)  : 2.0148
N°19 (Med: 65, 105, 148, 51)   : 1.7135  |  N°20 (Med: 31, 41, 33, 63)    : 1.1215

[>] Minimum Index Value : Sample Nr 2

Calculation time : 9.81000000000006 sec.
```

These informations can be hidden with the function *suppressMessages()*.

Object The function gives as output an object with the characteristic values of clustering.To get the dissimilarity matrix, the parameter *with_diss* must be TRUE.
The following parameters are returned :

```
$ seq #the original sequences dataset
$ id.med #the identifier of the medoids
$ clusters #the resulting clustering
$ diss #the matrix of distance of size k*n
```

CLARANS

CLARANS function has been implemented but is less efficient than CLARA. Indeed CLARANS cannot be parallelized and the calculation time is more important than CLARA's. If we compare CLARA and CLARANS for the same amount of time, we get several better results for CLARA.

Call

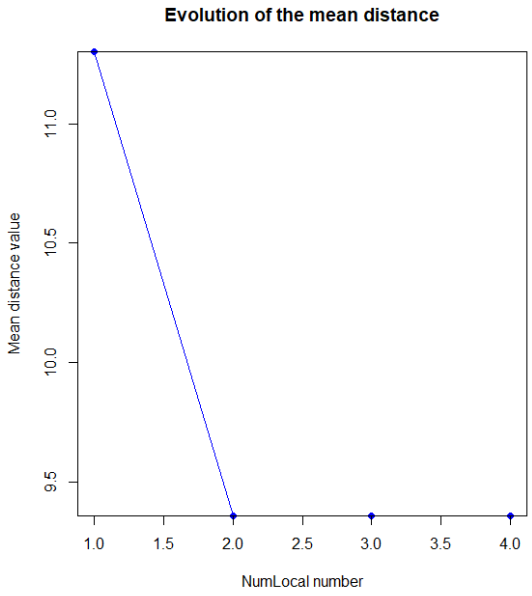
This function can still be used as follow :

```
my_clustering <- clarans_clust(full_matrix[1:100,],nb_cluster = 4, maxneighbours = 20,
+   numlocal = 4)
```

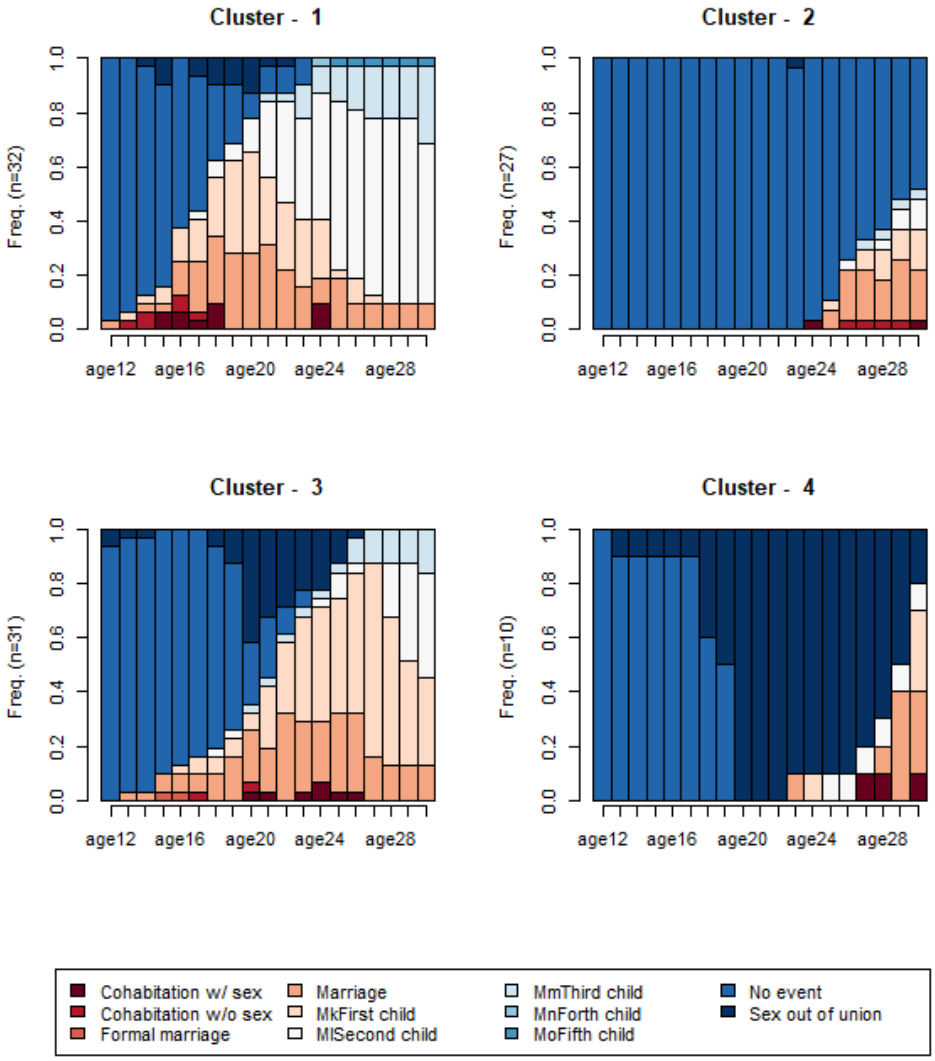
Outputs

Plots

- The first graph is the representation of the mean distance evolution :



- The second one shows the final clustering of the dataset :



Console outputs The function print for the users the details of the run and the calculation time needed :

```
CLARANS ALGORITHM

Table of mean distance per iteration

N°1 (Med: 19, 98, 4, 75)      : 9.2              N°2 (Med: 58, 97, 100, 91)    : 13.16
N°3 (Med: 22, 3, 55, 89)      : 10.82            N°4 (Med: 32, 97, 8, 63)      : 10.54

[>] Minimum Distance : iteration Nr 1

Calculation time : 8.72999999999956 sec.
```

These informations can be hidden with the function *suppressMessages()*.

Object The function gives as output an object with the characteristic values of clustering.
The following parameters are returned :

```
$ seq #the original sequences dataset
$ id.med #the identifier of the medoids
$ clusters #the resulting clustering
$ diss #the matrix of distance of size k*n
```

ROBUST FUZZY C-MEDOIDS

This function comes from the fuzzy algorithm family and provide a fuzzy clustering based on samples from the big dataset. This variant applies a fuzzy clustering on each subset and takes the most popular objects as medoids (determined with the membership matrix). For this algorithm, the objective function to minimise is :

$$\sum_{x_j \in S} u_{i,j}^m d(med, x_j)$$

where x_j is an object in the dataset without outliers S, i is the cluster for which a medoid is being sought, and med is the medoid tested.

Warning this function has not been fully tested.

Call

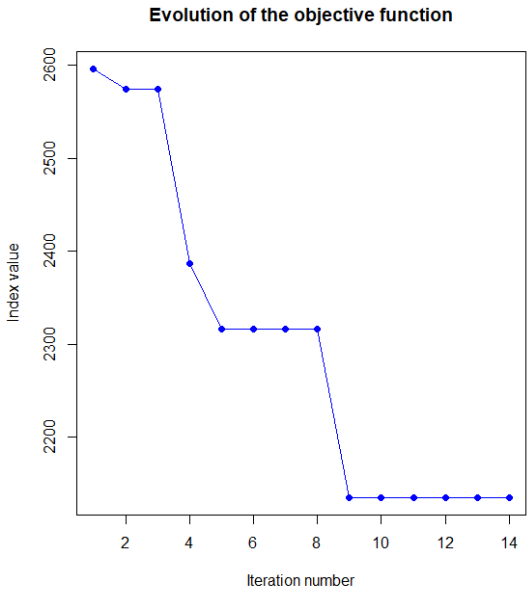
This function can be used as follow :

```
my_clustering <- fuzzy_clust(full_matrix_sample,nb_sample = 14, size_sample = 150,
+   plot = TRUE, threshold = 7, max_iter = 10, p=5)
```

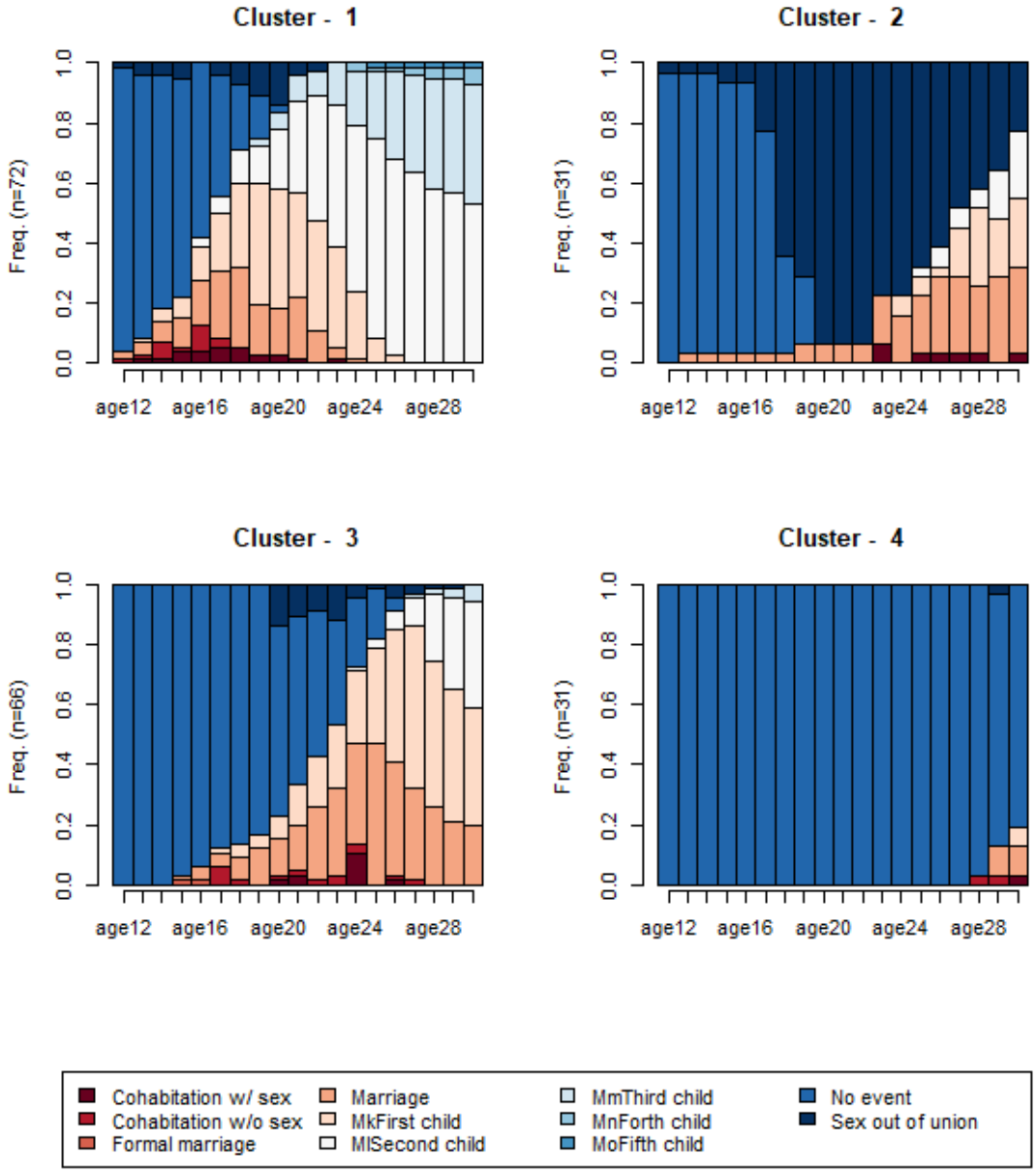
Outputs

Plots

- The first graph is the representation of the objective function evolution :



- The second one shows the final clustering of the dataset :



Console outputs The function print for the users the details of the run and the calculation time needed :

```
ROBUST FUZZY C-MEDOIDS ALGORITHM

Table of objective function values

N°1  (Med: 358, 168, 224, 32)  : 2596.0446  |  N°2  (Med: 10, 242, 217, 192)  : 2574.8131
N°3  (Med: 32, 284, 209, 63)   : 2719.9231  |  N°4  (Med: 288, 168, 65, 63)   : 2386.7981
N°5  (Med: 284, 1, 315, 233)   : 2316.6172  |  N°6  (Med: 205, 360, 55, 296)  : 2360.4724
N°7  (Med: 281, 226, 220, 22)  : 2400.5756  |  N°8  (Med: 320, 264, 17, 182)  : 2446.2271
N°9  (Med: 242, 397, 226, 238) : 2135.2379  |  N°10 (Med: 119, 224, 55, 41)   : 2573.7429
N°11 (Med: 146, 54, 96, 387)   : 3160.2132  |  N°12 (Med: 360, 64, 96, 209)   : 2988.3280
N°13 (Med: 105, 168, 233, 189) : 2354.0347  |  N°14 (Med: 189, 209, 171, 224) : 2140.7919

[>] Minimum Index for Sample Nr 9
```

```
Calcul time : 8.10999999999967 sec.
```

These informations can be hidden with the function *suppressMessages()*.

Object The function gives as output an object with the characteristic values of clustering.
The following parameters are returned :

```
$ seq #the original sequences dataset
$ id.med #the identifier of the medoids
$ clusters #the resulting clustering
$ diss #distance matrix
$ harm_value #harm value to adjust the threshold
$ nb_iter #number of iterations needed to find the medoids
$ membership #membership matrix
$ q #value of the index for each iteration
```