An R Code for Implementing Non-hierarchical Algorithm for Clustering of Probability Density Functions

Ngoc Diem TRAN [2], *Tom VINANT* [3], *Théo Marc COLOMBANI* [3], *Kieu Diem HO* [1,2,*]

[1] Division of Computational Mathematics and Engineering, Institute for Computational Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[2] Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[3] Statistics and Computer Science, Polytech Lille, University of Lille, Lille, France

*Corresponding Author: Kieu Diem HO (email: hokieudiem@tdtu.edu.vn)
(Received: 28-June-2018; accepted: 22-September-2018; published: 31-October-2018)
DOI: http://dx.doi.org/10.25073/jaec.201823.194

Abstract. *This paper aims to present a code for implementation of non-hierarchical algorithm to cluster probability density functions in one dimension for the first time in R environment. The structure of code consists of 2 primary steps: executing the main clustering algorithm and evaluating the clustering quality. The code is validated on one simulated data set and two applications. The numerical results obtained are highly compatible with that on MATLAB software regarding computational time. Notably, the code mainly serves for educational purpose and desires to extend the availability of algorithm in several environments so as having multiple choices for whom interested in clustering.*

Keywords

Crisp Clustering, Non-Hierarchical Algorithm, Probability Density Problem, R Code.

1. INTRODUCTION

R is a free, open-source implementation of the S programming language and computing environment for users. It is firstly developed in 1996 by professors Ross Ihaka and Robert Gentleman of the University of Auckland in New Zealand [1]. From time to time, a lot of upgraded versions are made in order to enhance its user library, more detail can be found at https://cran.r-project.org/. Due to aforementioned features, R does not require any purchased fee unlike other commercial software such as MATLAB or STATA. Moreover, it is also supported graphical presentation of data sets, built-in mechanisms for organizing data together with numerous available packages for users [2]. Therefore, it is extremely easy to use without good programming skills and it has been attracted a lot of attentions of experts in different fields, especially in statistics. To the best of our knowledge, R is currently one of the most well-known statistical software around the world as well as more and more packages are built to increase the diversity of R library [3]. Needless to say, R is an ideal environment for who want to contribute their humble work to other users to serve various purposes, educating and researching in particular.

Clustering is usually considered as grouping objects such that objects in the same cluster are similar as much as possible and differs from objects in other clusters. These days, it has been popular as an unsupervised recognition algorithm to explore the internal structure of data [4]. Therefore, more researchers are devoted to this field as a result. Concerning R packages related to clustering, a lot of works can be considered as follows. For instance, in 2006, Chris Fraley and Adrian E. Raftery made an enhancement for "MCLUST" package to serve for normal mixture modeling and model-based clustering. Simultaneously, Ryota Suzuki and Hidetoshi Shimodaira also contributed an R package so-called "Pvclust" to measure the uncertainty in hierarchical clustering analysis using the bootstrap resampling methods [5]. In 2007, Lokesh Kumar and Matthias Futschik offered a package called "Mfuzz" for fuzzy clustering of microarray data aiming at overcoming drawbacks of hard clustering in gene area [6]. Then, in 2011, another clustering-related package called "clValid" was released by Guy Brock at el. in order to evaluate the clustering result, concerning internal, stability and biological measures, respectively [7]. Besides, the work of Daniel Mullner in 2013 with a package so-called "fastcluster" based on C^{++} language boost performance of current clustering algorithms in R and Python both [8]. Further, in 2015, Tal Galili contributed a package to visualize, adjust and compare trees of hierarchical method [9]. Therefore, it can be noticed that the clustering field has gained a lot of attention from programmers year on year.

Nevertheless, one common point for these works is that they just consider the discrete object regardless probability density functions (PDFs). Meanwhile, there are two primary objects in clustering, discrete element and probability density function (PDF). Although the former one is more simple and convenient to handle due to less complex preprocessing data required, it still has the drawback that it could not represent the whole data by only one point. In contrast, in spite of few initial preprocessing, the later one is more advantageous since it is able to fully demonstrate character of the whole data [4]. Especially, this benefit is even more crucial in such a digital era nowadays. Thus, to fulfill the aforementioned research gaps as well as contributing more work in the field of clustering for PDFs, this paper will take into account of probability density functions in crisp clustering problem with number of clusters known in advance.

From what has been stated above in this paper, we propose an optimized code for non-hierarchical algorithm-based clustering of PDFs in R environment. Some interesting points can be enumerated as follows. Firstly, a code concerning the performance of clustering for PDFs is suggested in R software for the first time. In addition, one more extra function are also supplemented to validate the clustering quality. Next, we applied the code to execute some simulated data sets to confirm the accuracy of the code prior to giving an excited real example as an application. Finally, we extended the code for the input which is the object image instead of digital data. The achieved numerical result reveals a superior performance compared with that in MATLAB software in terms of computational time. More importantly, the code is always available for whom interested in clustering of PDFs or using this to serve education purpose.

The remaining structure of the paper is organized as follows. Section 2 briefly introduces some related basic knowledge. Section 3 outlines the main algorithm in the paper-non-hierarchical algorithm. Next, section 4 presents how to implement main functions of the code. Section 5 then performs some numerical examples to check the code in addition to comparing performance in MATLAB. Finally, Section 6 summarizes few conclusions of the whole work along with the future direction.

2. PRELIMINARY

2.1. Estimating PDFs from discrete elements

In reality, almost data are presented under forms of discrete elements, hence, we need to estimate the PDFs before making applications. There are many non-parametric methods, as also parametric ones to estimate PDFs. Among them, kernel method is one of the most popular ones in reality [10]. Its concept is shown briefly as follows.

Let $x_1, x_2, ..., x_N$ is N discrete elements (N observations) including n dimensions (n variables) which are employed to estimate PDFs. The kernel formula is presented as follows:

$$\hat{f}(x) = \frac{1}{Nh_1h_2...h_n} \sum_{i=1}^{N} \prod_{j=1}^{n} K_j \left(\frac{x - x_j}{h_j} \right), \quad (1)$$

where $h_j = \left(\frac{4}{N(n+2)} \right)^{\frac{1}{n+4}} \times \sigma_j$ is a bandwidth parameter for the jth variable; $K_j(\cdot)$ is a kernel function of the jth variable, which is usually Gaussian, Epanechnikov, Biweight, etc.

Bandwidth parameter and the type of kernel function have important roles in the estimation. There are many opinions to select a bandwidth parameter, but the optimal selection has not been found. In this paper, the bandwidth parameter is chosen based on the concept of Scott [11] and the kernel function is the Gaussian, where $\sigma_j = \sqrt{\frac{\sum_{i=1}^{N} (x_{ij} - \bar{x}_j)^2}{N-1}}$ is sample standard deviation for the jth variable.

It is noticed that estimating PDFs in one dimension is considered in this paper.

The clustering problem of probability density functions is defined as follows:

Definition 1. Let F be a set of PDFs, $F = \{f_1(x), f_2(x), ..., f_n(x)\}$, $n > 2$ and k be the number of clusters given in advance. The requirement of the crisp clustering problem is to separate these PDFs according to value of k into clusters $C_1, \cdots, C_k$ such that

i) $\sum_{i=1}^{k} \#C_i = n$, where $\#C_i$ is the number of PDFs in cluster i.

ii) $C_i \cap C_j = \emptyset$, $i \neq j$, $i = \overline{1,k}$, $j = \overline{1,k}$.

2.2. Evaluating similarity between PDFs

In clustering problem, how to determine the similarity between PDFs is extremely crucial since it partly decides the clustering result [12]. Although numerous similarity measures are proposed for discrete elements, there is a restricted number for PDFs. To the best of our knowledge, L^1 distance and cluster width are one of the most popular. Thus, in this paper, we select the cluster width to measure the similarity between PDFs. Its definition is briefly expressed as follows [13].

Definition 2. Given n PDFs $f_1(x), f_2(x), \ldots, f_n(x), (n \geqslant 2)$ defined on R^m, let $f_{\max}(x) = \max\{f(x)_1, f_2(x), \ldots, f_n(x)\}$, the cluster width is defined as

i) $n = 2$

$$w(f_1, f_2) \equiv \frac{\|f_1 - f_2\|_1}{2} = \int_{R^m} f_{\max}(x)\, dx - 1 \quad (2)$$

ii) $n > 2$

$$w(f_1, f_2, \ldots, f_n) \equiv \|f_1, f_2, \ldots, f_n\|_1 = \int_{R^m} f_{\max}(x)\, dx - 1. \quad (3)$$

Definition 3. Let $g, (g_1, g_2, ..., g_n), (f_1, f_2, ..., f_m)$ are probability of density functions, we define the cluster width between a PDF and a set of PDFs (cluster) is $w[g \cup \{(f_1, f_2, ..., f_m)\}]$ and the cluster width between two sets of PDFs (two clusters) is $w[\{g_1, g_2, ..., g_n\} \cup \{f_1, f_2, ..., f_m\}]$.

2.3. SF index- an internal validity measure index

After a partition is deduced from a clustering algorithm, one needs to evaluate the goodness of partition. Normally, the internal validity measure indexes intend to perform that purpose. For internal measure index, the SF stated in [14] is employed to assess the internal structure of cluster. In detail, this measure is defined below.

$$SF = \frac{\sum_{i=1}^{k} \sum_{f \in C_i} \|f - fv_i\|^2}{n\min_{i \neq j}\left(\|fv_i - fv_j\|^2\right)}, \quad (4)$$

where $\|fv_i - fv_j\|$ is the L^1 distance between representing PDFs of the clusters C_i and C_j;

$fv_j = \frac{\sum\limits_{f_i \in C_j} f_i}{n_j}$ is the representing probability density function for each cluster with n_j being the number of PDFs in the cluster C_j.

From Eq. (4), we see that SF considers not only the compactness among PDFs in one cluster but also concerns the separation between clusters. Therefore, it is reasonable to employ SF in this paper as an internal measure index. The smallest value of SF, the most valid optimal partition indicates [14].

2.4. Extracting image features

In this section, we will present shortly how to cluster for image objects based on their features. Initially, we will read color feature from image pixels into R software. From the pixel distribution of Grayscale or RGB scale, we can construct one-dimensional or multi-dimensional PDF representing for each image. These PDFs will be the input for the employed algorithm to tackle subsequent works. Besides, all processing steps like reading image and presenting an image in one-dimensional or multi-dimensional spaces are performed in R software by some available packages.

3. NON-HIERARCHICAL ALGORITHM

The non-hierarchical algorithm firstly proposed by T.V.Van and Pham-Gia [13] presents a new approach for clustering of PDFs with prior number of clusters. The original idea is inspired from the well-known k-means algorithm in clustering of discrete elements. That is, from a known number of clusters, an arbitrarily initial partition is created to assign each probability density function (PDF) into each cluster. These PDFs will then reallocate to cluster such that cluster width is the minimum.

Given n PDFs $f_1, f_2, \ldots, f_n \, (n > 2)$ being separated into k clusters $C_1, C_2, \ldots, C_k$ such that $\sum\limits_{i=1}^{k} \#C_i = n$, denoting that $C_i^{(t)}$ is the cluster C_i, $i = \overline{1,k}$ at iteration tth, the non-hierarchical algorithm is shown as follows.

Step 1. Partitioning n PDFs into k random clusters, we have initial clusters at the 1st iteration $C_1^{(1)}, C_2^{(1)}, \ldots, C_k^{(1)}$.

Step 2. For a specific f_j, denoting $w\left(f_j \cup C_i^{(1)}\right) = \left\| f_j \cup C_i^{(1)} \right\|_1$, $i = \overline{1,k}$, $j = \overline{1,n}$ is the cluster width of set obtained by assigning f_j to clusters $C_1^{(1)}$, computed according to Eq. (2).

Consider just $f_j \in C_h^{(1)}$, there are two possible cases as follows:

a) If $w\left(f_j \cup C_h^{(1)}\right) = \min_{i=1,\ldots,k} \left\{ w\left(f_j \cup C_i^{(1)}\right) \right\}$, keep f_j in cluster $C_h^{(1)}$.

b) If there exist another $C_s^{(1)}$ such that $w\left(f_j \cup C_s^{(1)}\right) = \min_{i=1,\ldots,k} \left\{ w\left(f_j \cup C_i^{(1)}\right) \right\}$, the f_j will move to the cluster $C_s^{(1)}$ to establish new cluster in the 2nd iteration: $C_s^{(2)}$. Simultaneously, the old cluster $C_h^{(1)}$ will detached f_j to form the new one $C_h^{(2)}$ in next step.

Now, the new partition is formed: $C_1^{(2)}, C_2^{(2)}, \ldots, C_k^{(2)}$ where f_j has been moved to the right cluster with the minimum cluster width.

Step 3. Repeat Step 2 m times until no more change for each PDF in each cluster, that means all k clusters at the mth iteration $C_1^{(m)}, C_2^{(m)}, \ldots, C_k^{(m)}$ satisfying $w\left(f_j \cup C_s^{(m)}\right) = \min_{i=1,\ldots,k} \left\{ w\left(f_j \cup C_i^{(m)}\right) \right\}$.

Figure 1 is the diagram demonstrating for the mentioned process.

4. IMPLEMENTATION IN R SOFTWARE

4.1. Main program

The main program starting from line 1 to 149 aims to perform the non-hierarchical algorithm in clustering for PDFs. It is called from command window of R by the line

$$\text{cluster}(f, k, x) \qquad (5)$$

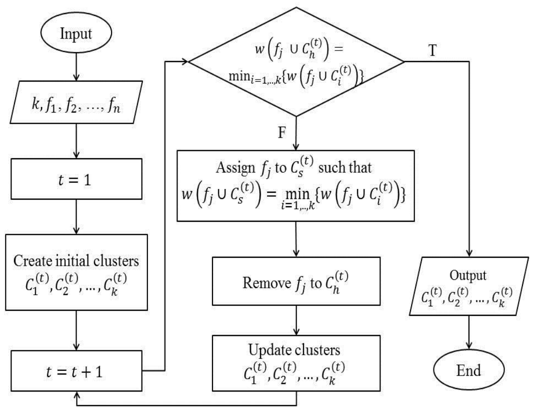
Input
$k, f_1, f_2, \ldots, f_n$
$t = 1$
Create initial clusters $C_1^{(t)}, C_2^{(t)}, \ldots, C_k^{(t)}$
$t = t + 1$
$w\left(f_j \cup C_h^{(t)}\right) = \min_{i=1,\ldots,k}\{w\left(f_j \cup C_i^{(t)}\right)\}$
T
F
Assign f_j to $C_s^{(t)}$ such that $w\left(f_j \cup C_s^{(t)}\right) = \min_{i=1,\ldots,k}\{w\left(f_j \cup C_i^{(t)}\right)\}$
Remove f_j to $C_h^{(t)}$
Update clusters $C_1^{(t)}, C_2^{(t)}, \ldots, C_k^{(t)}$
Output $C_1^{(t)}, C_2^{(t)}, \ldots, C_k^{(t)}$
End

Fig. 1: Flowchart demonstrates process of non-hierarchical algorithm.

where

f is the one-dimension PDFs to be grouped. Normally, f is saved under the form of matrix with each column representing for each PDF;

k is the number of clusters known in advance. Usually, the value of k ranges from 2 to square root of number of PDFs;

x is value used for evaluating f on a define domain.

The non-hierarchical algorithm starts by creating and printing the initial partition matrix (lines 7-9). Herein, we build a subroutine called "createU" to perform this task. After the partition is established, the next step assigns the input PDFs into clusters in which PDFs belong to (lines 12-17). Subsequently, computing cluster width between each PDF with all clusters is executed (lines 20-28). This step intends to calculate the similarity level of PDF in each cluster to be reference for later changes. Before going to the while loop for updating the clusters, some variables are created firstly. For example, m and d are respectively used to count the number of elements and record the minimum cluster width when observed PDF does not satisfy minimum cluster width between it and cluster it belonging to. *Unew* is employed to update the initial partition matrix. Then, checking initial partition matrix **U**, if **U** satisfies the condition, the algorithm will stop and output the **U** as the final partition (lines 49-51). Otherwise, updating matrix **U** is performed starting from line 52. The while loop for updating partition matrix will suspend if m is equal to 0 meaning all PDFs satisfy condition that the cluster width of a PDF to cluster, which it is belonging to, is minimum.

4.2. Creating initial partition matrix

The initial partition matrix is created by subroutine so-called "createU" (lines 151-173) to randomly make a partition before performing the non-hierarchical algorithm. This function can be used by following command

$$createU(k, n_{\text{pdf}})$$

where k is the number of clusters and n_{pdf} is the total number of PDFs initially.

Firstly, variable *comp* is created to record the number of PDFs in each cluster (line 153). The while loop then will generate a partition matrix until none of cluster is empty. The variable B is used to create a row column where each element represents the label of cluster each PDF belonging to (lines 157-158). Next, lines 159-163 establish the matrix **U** based on vector B. Note that here *runif*$(1*n_{\text{pdf}}, .5, k+.5)$ intends to generate a row vector with uniform distribution so that each PDF is able to assign to one cluster with equal probability.

4.3. Computing cluster width

The cluster width is calculated numerically based on Monte-Carlo method. The subroutine WidthCluster(f, x) is responsible for this task from lines 174-180. It is noticed that f in command *((sum(rowMaxs(f, na.rm = FALSE))/length(x))*(x[length(x)] - x[1]))* must contain two PDFs or more.

4.4. Validate clustering result by SF-internal validity measure index

After clustering PDFs, one needs to evaluate these established clusters. The SF function will perform this task by subroutine SF (f, $\mathbf{U}$, x), where inputs already mentioned (lines 181-227). Note that variable *id* is a list to record PDFs of each cluster and variable *ni* is a vector storing number of PDFs of each cluster. After computing all these main variables, calculating representing PDFs is considered (lines 193-206). One subroutine called L^1 (lines 228-233) is employed to compute distance between PDFs.

5. NUMERICAL EXAMPLE

In this section, we employ the code written in R to perform one simulated data set and two applications. The first one is seven PDFs having normal distribution separated into 3 clusters. The numerical result of this data set will be compared with that in MATLAB to have some evaluations, especially the computational time. Then, two applications are executed, one is data about satisfaction level of student at Ton Duc thang University, another is traffic image. For real data, the nominal partition is not determined yet so that we just assess in terms of computational time, SF and not comparing with other software. Moreover, for each data set, the performance is repeated 30 independent times to assure the stability and accuracy. The final numerical results are computed by average of these 30 times. The detail of each numerical result will be presented later.

5.1. Simulated data

For this case, we consider a Benchmark dataset in order to validate the accuracy of the code. That is seven univariate normal probability density functions (PDFs) firstly proposed in [13]. The structure of data is well-separated and less overlapping leading to reducing the complexity for clustering mission. Therefore, it is used to test performance of a novel algorithm or algorithm recoded in other programming environment like R. The detail of estimated parameters can be referred to [13]. From these parameters, seven PDFs are estimated and demonstrated in Fig. 1. Clearly, from Figs. 2-3 clusters could be defined as

$$C_1 = \{f_1, f_4\}, C_2 = \{f_2, f_5, f_7\}, C_3 = \{f_3, f_6\}.$$

The numerical result is shown in Table 1 and Fig. 2. It is clear that both R and MATLAB produce precise results compared with the nominal partition with lowest SF 0.050. Nevertheless, the code written on R is more rapid than in MATLAB since it just takes 0.016 seconds in average in R instead of 0.038 in MATLAB. Therefore, writing code on R not only provides a new environment for users to experience but also boosts the performance of non-hierarchical algorithm in this case.

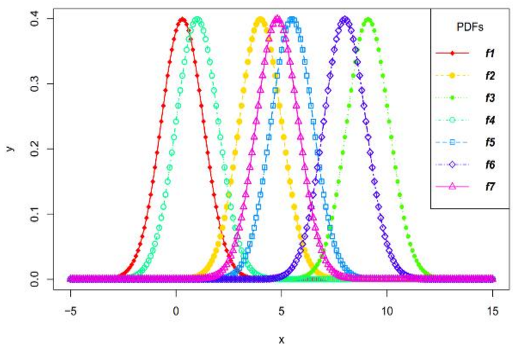

Fig. 2: Seven PDFs having univariate normal distribution in example 1 before clustering.

Table 1. Comparison of performance of non-hierarchical algorithm for seven normal PDFs in R and MATLAB software.

Software	SF	Computational Time (seconds)
R	0.050	0.016
MATLAB	0.050	0.038

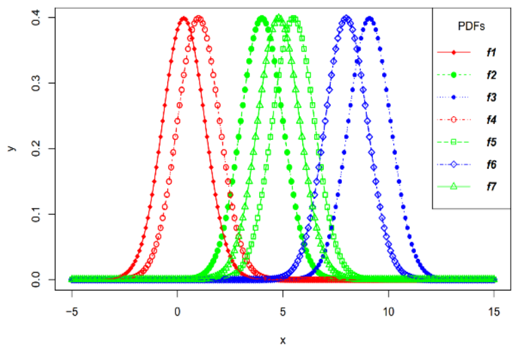

Fig. 3: The clustering result of seven PDFs in example 1 by non-hierarchical algorithm in R.

5.2. Application to clustering student satisfaction level at Ton Duc Thang University

In this example, we take a real data to run by R code. The data considers statistics of student satisfaction levels at Ton Duc Thang University (TDTU) toward lecture's teaching performance. The data consists of 16 departments as follows.

1. Faculty of Foreign Languages
2. Faculty of Industrial Fine Arts
3. Faculty of Accounting
4. Faculty of Drupal Social Sciences and Humanities
5. Faculty of Electrical Electronics Engineering
6. Faculty of Information Technology
7. Faculty of Applied Sciences
8. Faculty of Business Administration
9. Faculty of Civil Engineering
10. Faculty of Environment and Labor Safety
11. Faculty of Labor Relations and Trade Unions
12. Faculty of Finance and Banking
13. Faculty of Mathematics and Statistics
14. Faculty of Sport Science
15. Faculty of Law
16. Faculty of Pharmacy

Herein, the data is preprocessed to remain valid values and then computing the average satisfaction score of each student to serve for estimating the PDFs.

PDFs estimated from data are illustrated in Fig. 4. One can see that the PDFs are pretty overlapping so that hardly can algorithm cluster correctly all PDFs. Furthermore, since the number of clusters is undetermined, hence some numbers are suggested to survey, including 2, 3 and 4 clusters. The most appropriate result is measured in terms of SF index. If the SF value is smaller, it means that the partition is better, and the number of clusters will be taken based on that.

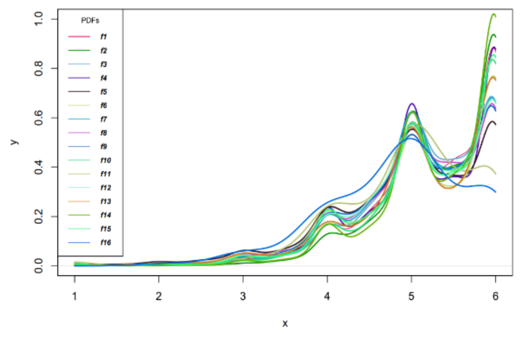

Fig. 4: Sixteen PDFs estimated by sixteen faculties in example 2 before clustering.

All numerical results corresponding to suggested number of clusters are listed in Table 2.

Table 2. The result for three cases of k uses non - hierarchical algorithm for sixteen PDFs in R software.

	k =2	k =3	k =4
Min of SF Index	0.154	0.332	0.347
Mean of Time (seconds)	0.112	0.285	0.295

From the Table 2, it can be said that the case $k = 2$ has the smallest SF Index, so that we will divide sixteen faculties into two clusters:

$$C_1 = \{f_5, f_{11}, f_{16}\},$$
$$C_2 = \left\{ \begin{array}{l} f_1, f_2, f_3, f_4, f_6, f_7, \\ f_8, f_9, f_{10}, f_{12}, f_{13}, f_{14}, f_{15} \end{array} \right\}.$$

Based on the clustering result and comments from the students on the lecturer during the survey, some main discussions are listed as follows:

- The Cluster 1 (C_1) includes faculties: Electrical Electronics Engineering, Labor Relations and Trade Unions, Pharmacy have average satisfaction scores from 3 to below 5 (it means from "slightly dissatisfaction" level to "quite satisfaction" one) more than the rest of the faculties. That is because students have a lot of dissatisfied opinions with faculty members from teaching methods, teaching materials, student interaction, enthusiasm, activity more than the rest of the faculties.
- The Cluster 2 (C_2) consisting of remaining faculties has an average satisfaction score from 5 to 6 (it means from "satisfaction" level to "very satisfaction" one) higher than Cluster 1. Although faculty members belonging to C_2 received some unexpected comments, almost comments are positive such as good and speed teaching method, enthusiastic lectures. As a result, students give a pretty high level of satisfaction for this group (C_2).

From the above comments, it points out that each department needs to take measures to improve the bad points, promote the strengths of lecturers to be better teaching so that students is getting more and more satisfaction.

5.3. Application to clustering traffic images

In this example, we would like to demonstrate an example for image object in implementation section. Specifically, there are 121 real images size 1920 × 1080 pixels extracted from a short video in front of TDTU-Nguyen Huu Tho street at night. Some sample images are represented in Fig. 6. Firstly, these images are digitized to estimate corresponding PDFs. Then, these PDFs are considered to be the input of the main algorithm for clustering. Based on some prior knowledge, some number of clusters are proposed, including 2, 3 and 4 clusters. SF-based assessment will be applied similarly to the previous example to select the most suitable partition and number of clusters.

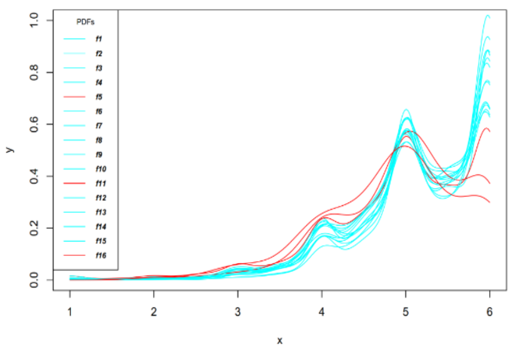

Fig. 5: The clustering result of sixteen PDFs in example 2 by non-hierarchical algorithm in R.

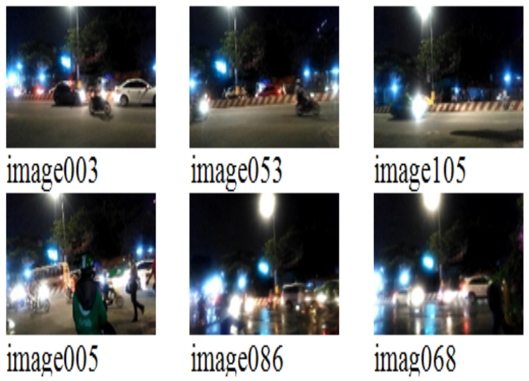

Fig. 6: Some traffic images extracted from a short video on Nguyen Huu Tho Street at night.

The PDFs estimated are demonstrated in Fig. 7. It is obviously that almost PDFs are significantly overlapped resulting in challenging the non-hierarchical algorithm to deduce a good partition.

The numerical result is shown in Table 3. As can be seen from the table that the case $k = 2$

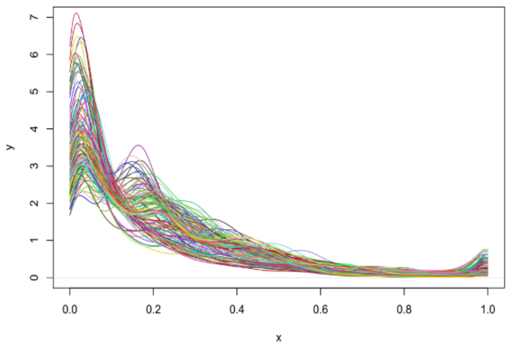

Fig. 7: 121 PDFs estimated by 121 traffic images on Nguyen Huu Tho Street at night in example 3 before clustering.

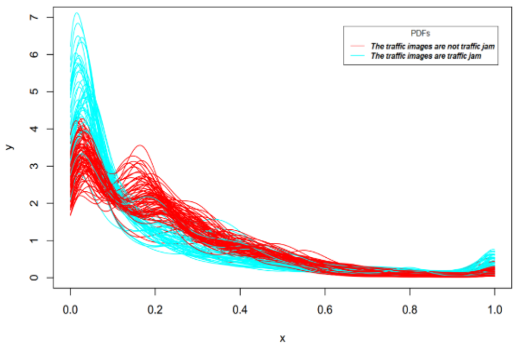

Fig. 8: The clustering result of 121 PDFs in example 3 by non-hierarchical algorithm in R.

has the smallest SF index. This shows that at every moment, there are just two main state traffic: traffic jams and no traffic jams. Other states are not really clear in this situation. This result also reveals an useful application of the algorithm in real problem. Figure 8 is set of probability density functions extracted from images, so that we can see that the group of red probability functions are traffic images classified as cluster of not traffic jams and the group of light blue probability are traffic images classified as cluster of traffic jams.

Table 3. The results for three cases of k using non - hierarchical algorithm for 121 PDFs in Example 3 in R software

	k =2	$k = 3$	$k = 4$
Min of SF Index	0.267	0.558	0.893
Mean of Time (seconds)	2.273	4.686	13.546

6. CONCLUSION

This paper provides an R code of non-hierarchical algorithm using for clustering problem of probability density functions. This aims to extend the range of this algorithm as well as offering more options for whom interested in clustering for PDFs. By advantages of R software, this would bring convenience for researchers in addition to educators. Overall, the structure of the code and how to implement it are fully presented. Moreover, some simulations and applications are also used to validate the accuracy plus the computational time of the code. The result shows that the code in R is superior than that in MATLAB regarding the computational time. Furthermore, the source code is completely shown in the Appendix section so that one can refer for more detail.

Nevertheless, the provided code in this paper is just presented in one dimension. Therefore, two or more dimensions should be added in order to expand the code as well as develop the applications of the non-hierarchical method. This would be a promising direction in our future work.

References

[1] Ihaka, R., & Gentleman, R. (1996). R: a language for data analysis and graphics. Journal of computational and graphical statistics, 5(3), 299-314.

[2] Fox, J., & Andersen, R. (2005). Using the R statistical computing environment to teach social statistics courses. Department of Sociology, McMaster University, 2-4.

[3] Muenchen, R. A. (2014). The popularity of data analysis software. no. April.

[4] Nguyen Trang, T., & Vo Van, T. (2017). Fuzzy clustering of probability density functions. Journal of Applied Statistics, 44(4), 583-601.

[5] Suzuki, R., & Shimodaira, H. (2006). Pvclust: an R package for assessing the uncertainty in hierarchical clustering. Bioinformatics, 22(12), 1540-1542.

[6] Kumar, L., & Futschik, M. E. (2007). Mfuzz: a software package for soft clustering of microarray data. Bioinformation, 2(1), 5.

[7] Brock, G., Pihur, V., Datta, S., & Datta, S. (2011). clValid, an R package for cluster validation. Journal of Statistical Software (Brock et al., March 2008).

[8] Müllner, D. (2013). fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. Journal of Statistical Software, 53(9), 1-18.

[9] Galili, T. (2015). dendextend: an R package for visualizing, adjusting and comparing trees of hierarchical clustering. Bioinformatics, 31(22), 3718-3720.

[10] Vo Van, T. (2017). L 1-distance and classification problem by Bayesian method. Journal of Applied Statistics, 44(3), 385-401.

[11] Scott, D. W. (2015). Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons.

[12] Ho-Kieu, D., Vo-Duy, T., Vo Van, T., & Nguyen Trang, T. (2018). A Differential Evolution-Based Clustering for Probability Density Functions. IEEE Access, 6, 41325-41336.

[13] Vo Van, T., & Pham-Gia, T. (2010). Clustering probability distributions. Journal of Applied Statistics, 37(11), 1891-1910.

[14] Vo Van, T., Nguyen Trang, T., & Che-Ngoc, H. (2016, March). Clustering for probability density functions based on Genetic Algorithm. In Applied Mathematics in Engineering and Reliability, Proceedings of the 1st International Conference on Applied Mathematics in Engineering and Reliability (Ho Chi Minh City, Vietnam, May 2016) (pp. 51-57).

About Authors

Tran Thi Ngoc DIEM received the Bachelor degree in Statistics at Ton Duc Thang University, Ho Chi Minh City, Vietnam in 2018. Her research interests include statistics, machine learning, big data and data analysis.

Tom VINANT is a student from a French engineering school, Polytech Lille, in Statistics and Computer Science Department for 3 years. He has been working on this thesis subject during a two months internship at Institute for Computational Science (INCOS), Ton Duc Thang University, Ho Chi Minh City, VietNam.

Théo Marc COLOMBANI is a student from the French engineering school Polytech Lille. He has been studying in Statistics and Computer Science Department for 3 years. He has been working on this subject at Institute for Computational Science (INCOS), Ton Duc Thang University, Ho Chi Minh City, VietNam and supervised by Thao Trang Nguyen.

Kieu Diem HO received the B.Sc degree in Applied Mathematics at Can Tho University, Can Tho City, Vietnam and the M.Sc degree also in Applied Mathematics at University of Orléans, France. Her research interests include unsupervised recognition, supervise recognition, particularly analyzing and applying techniques to discover the potential information in data science. She was an Assistant Researcher at Institute for Computational Science (INCOS), Ton Duc Thang University, Ho Chi Minh city, Vietnam. Currently, she is a doctoral student in computer science at Polytech Tours, Tours University, France.

APPENDIX - R CODE FOR IMPLEMENTATION OF NON-HIERARCHICAL ALGORITHM

#Function to clustering of PDFs based on non-hierarchical method

```
cluster <- function(f, k, x)
{
#Randomly create an initial partition matrix
iter <- 0
U <- createU(k, $n_{\text{pdf}}$)
print("Initial matrix")
print(U)
#Attach the value of the element to the cluster that it belongs to
C <- vector('list', k)
for (i in 1:k)
{
id = which(U[i,] %in% c(1))
C[[i]] = f[,id]
}
#Calculates the cluster width of each element to the clusters
W <- matrix(0, k, ncol(f))
for (j in 1:ncol(f))
{
for (i in 1:k)
{
W[i,j] =
ClusterWidth(cbind(f[,j], C[[i]]), x)
}
}
m <- 0
Unew <- U
j <- 1
d <- c()
#Start while loop
while (j < ncol(f)+.1)
{
#Identify which elements belong to which cluster
r = which(U[,j] %in% c(1))
#Identify the element does not satisfy the cluster that it belongs to
if (W[r,j] != min(W[,j]))
{
temp1 = min(W[,j])
d = c(d, temp1)
m = m + 1
} #End if loop
j = j + 1
} #End while loop
if (length(d) == 0)
{
U = Unew #Output clustering results
} else {
dmin <- min(d)
#Update the partition matrix
j <- 1
while (j < ncol(f)+.1)
{
#Identify which elements belong to which cluster
r = which(U[,j] %in% c(1))
for (i in 1:k)
{
if ((W[r,j] != min(W[,j]))&& (min(W[i,j]) == dmin))
{
Unew[r,j] = 0
Unew[i,j] = 1
}
}
```

```
j = j + 1
U=Unew
} #End while loop
print("Number of iterations")
iter = iter + 1
print(iter)
print(U)
#End While loop
####################
#Update process
while (m > 0)
{
print("Number of iterations")
iter = iter + 1
print(iter)
Cnew = vector('list', k)
for (i in 1:k)
{
idd = which(U[i,] %in% c(1))
Cnew[[i]] = f[,idd]
}
Wnew = matrix(0, k, ncol(f))
for (j in 1:ncol(f))
{
for (i in 1:k)
{
Wnew[i,j] = ClusterWidth(cbind(f[,j], Cnew[[i]]), x)
}
}
m <- 0
Unew <- U
j <- 1
dnew <- c()
while (j < ncol(f)+.1)
{
#Identifies which elements belong to which cluster
r = which(U[,j] %in% c(1))
#Identify the element does not satisfy the cluster it belongs to
if (Wnew[r,j] != min(Wnew[,j]))
{
temp3 = min(Wnew[,j])
dnew = c(dnew, temp3)
m = m + 1
}
j = j + 1
} #End while loop
if (length(dnew) == 0)
{
U = Unew #Output clustering results
} else {
dminnew <- min(dnew)
#Update the partition matrix
j <- 1
while (j < ncol(f)+.1)
{
#Identifies which elements belong to which cluster
r = which(U[,j] %in% c(1))
#End Identification
for (i in 1:k)
{
if ((Wnew[r,j] != min(Wnew[,j])) && (min(Wnew[i,j]) == dminnew))
{
Unew[r,j] = 0
Unew[i,j] = 1
}
}
j = j + 1
} #End while loop
U = Unew
print(U)
} #End If else the second
} #End While Loop from #Cluster
} #End If else the first time
U #Output clustering results
} #End Cluster function
# Function to create initial partition matrix
createU<-function(k, n_pdf)
{
comp <- rep(0, k)
iter <- 0
while (comp[which.min(comp)] == 0)
{
B <- runif(1*n_pdf, .5, k + .5)
B <- round(B)
U <- matrix(rep(0, k*n_pdf ), k, n_pdf )
for (i in 1:n_pdf)
{
U[B[i],i] = 1
}
for (i in 1:k)
{
temp = sum(U[i,])
comp[i] = temp
}
comp
iter = iter + 1
}
U
}
#Function to compute cluster width
WidthCluster <- function(f,x)
{
((sum(rowMaxs(f, na.rm = FALSE))/length(x))*(x[length(x)] - x[1])) - 1
}
#Function to compute SF index
SF <- function(f,U,x)
{
k <- nrow(U)
id <- list()
ni <- c()
for (i in 1:k)
{
id[[i]] = which(U[i,] %in% c(1))
temp1 = length(id[[i]])
ni = c(ni, temp1)
} #End for loop
#Compute representing probability density function
fv <- list()
for (i in 1:k)
{
idd = which(U[i,] %in% c(1))
if (ni[i] == 1)
{
fv[[i]] = f[,idd]
} else {
fv[[i]] = (1/ni[i])*(rowSums(f[,idd]))
}
}
fv
li <- c()
for (i in 1:k)
{
for (j in 1:ni[i])
```

```
{
temp2 = (L1(f[,id[[i]][j]], fv[[i]], x))^2
li = c(li, temp2)
}
}
di <- c()
for (i in 1:(k-1))
{
for (j in (i+1):k)
{
temp3 = (L1(fv[[i]], fv[[j]], x))^2
di = c(di, temp3)
}
}
SF <- ((1/ncol(f))*sum(li))/min(di)
SF
}
#Function to compute L1 distance
L1 <- function(f1, f2, x)
{
sum(abs(f1 - f2))/length(x)*(max(x) -
min(x))
}
```