algorithms

MDPI

Article

Simple K-Medoids Partitioning Algorithm for Mixed Variable Data

Weksi Budiaji [1,2,*] and Friedrich Leisch [2]

[1] Agribusiness Department, Sultan Ageng Tirtayasa University, Jl. Raya Jakarta Km. 4 Serang, Banten 42118, Indonesia
[2] Institute of Statistics, University of Natural Resources and Life Sciences, Peter Jordan Straße 82, 1180 Vienna, Austria
* Correspondence: budiaji@untirta.ac.id

Received: 15 July 2019; Accepted: 20 August 2019; Published: 24 August 2019

Abstract: A simple and fast k-medoids algorithm that updates medoids by minimizing the total distance within clusters has been developed. Although it is simple and fast, as its name suggests, it nonetheless has neglected local optima and empty clusters that may arise. With the distance as an input to the algorithm, a generalized distance function is developed to increase the variation of the distances, especially for a mixed variable dataset. The variation of the distances is a crucial part of a partitioning algorithm due to different distances producing different outcomes. The experimental results of the simple k-medoids algorithm produce consistently good performances in various settings of mixed variable data. It also has a high cluster accuracy compared to other distance-based partitioning algorithms for mixed variable data.

Keywords: cluster; distance; partitioning; k-medoids; mixed variable data

MSC: 62H30

1. Introduction

Cluster analysis is a vital exploratory tool in data structure investigation. Each object within a group is similar (homogeneous) to each other, and objects between groups are distinct (heterogeneous) from one another [1,2]. In distance-based clustering, a similarity quantification between objects is based on the distance; more similar objects have a closer distance and vice versa. After the distance is defined, either a hierarchical or partitioning algorithm is applied.

One of the popular partitioning algorithms is k-means [3], which is one of the top ten algorithms in data mining [4]. The k-means algorithm, however, is irrelevant when the data are mixed variable data because "means" as the center of the clusters (centroid) are unavailable and the Euclidean distance is not applicable. Then, a modified k-means called a k-prototype was developed to manage a mixed variable dataset [5]. The k-prototype is similar to the k-means algorithm with pseudo-means (prototypes) as the centroids. The most common practice for a mixed variable dataset is applying the partitioning around medoids (PAM) [2], which replaces the centroids with the medoids.

The other approach to calculate a distance for the mixed variable data is via a combined distance function of the numerical and categorical distances [5–8]. The combined distance function assumes an equal contribution of each individual variable. When the individual variables contribute differently in a subspace clustering domain, unequal weights in the Gower distance [9], which is the most common distance for a mixed variable dataset, can be applied [10]. However, the assumption of an equal contribution of each variable in a distance function is retained in this article because this assumption is valid for a regular clustering method.

After defining the distance for the mixed variable data, either the k-prototype or PAM algorithm is applied. The difference between the two algorithms relies on the cluster center definition. The k-prototype determines the centroid by the combinations of the means/median (numerical) and mode (binary/categorical) or proportional distribution (categorical) [5,6,11–14], depending on the distance function. Meanwhile, PAM takes the most centrally-located (medoid) one as the cluster center for any distance function. Thus, PAM as a medoid-based algorithm is more robust with respect to the cluster center definition than centroid-based algorithms.

K-medoids (KM) [15] and simple and fast k-medoids (SFKM) [16] as medoid-based algorithms have been proposed. The KM and SFKM algorithms differ in the initial medoid selection, where the former selects them arbitrarily, while the latter chooses via a specific formula. In the medoid updating step, on the other hand, they are very similar. Both algorithms operate within clusters only, like centroid updating in k-means, for the medoid updating. As a result of the within cluster medoid updating, local optima [17], as well as empty clusters [18] that are likely to arise are not addressed by these algorithms.

In this paper, by taking local optima and empty clusters into consideration, we propose a k-medoids algorithm that improves the performance of the SFKM algorithm (SFKM). In addition to that, we generalize a distance function, which is feasible for any numerical and categorical distance combination and its respective weight.

2. Related Works and the Proposed Method

2.1. K-Medoids Algorithms

A well-known choice of a medoid-based algorithm is the PAM algorithm [2]. The PAM algorithm consists of three steps: initial medoid selection, medoid swapping, and object labeling. The initial medoids are chosen randomly. Then, medoids are swapped with randomly-selected $n - k$ possible medoids, where n is the number of objects and k is the number of clusters.

Besides PAM, the k-medoids (KM) [15] and simple and fast k-medoids (SFKM) [16] algorithms have been developed. Both algorithms have restricted the medoid swapping step. It works within clusters only, i.e., similar to the centroid updating in k-means. However, they both differ in the initial medoid selection where the KM algorithm uses randomly-selected initial medoids. Meanwhile, SFKM selects the initial medoids using a set of ordered v_j. The value of v_j is defined as a standardized row sum or standardized column sum of the distance matrix that is calculated by:

$$v_j = \sum_{i=1}^{n} \frac{d_{ij}}{\sum_{l=1}^{n} d_{il}}, \quad j = 1, 2, 3, \ldots, n,$$

where d_{ij} is the distance between object i and object j. When the pre-determined number of clusters is k, the first k smallest set is selected from the ordered v_j as the initial medoids.

Due to the similarity to the k-means centroid updating, the KM and SFKM algorithms suffer from possible local optima [17] and empty cluster [18] problems. The result of the partition in the k-means greatly depends on the initialization step. The work in [17] listed and compared 12 initialization strategies in order to obtain a suitable final partition. For general practice, the procedure of multiple random initial centroids produces a very good partition compared to more complicated methods. In addition, when two or more initial centroids are equal, an empty cluster occurs. It can be solved by preserving the label of one of the initial centroids [18].

For the k-medoids algorithm, on the other hand, a ranked k-medoids [19] has been introduced as a way out for the local optima. Instead of using the original distance between objects, it applies a new asymmetric distance based on ranks. The asymmetric rank distance can be calculated by either a row or column direction. Despite its way out of the local optima problem, this algorithm transforms the original distance into an asymmetric rank distance such that a tie rank of two objects having an equal

distance can arise. However, it does not discuss how a tie rank is treated, whereas it can be a source of an empty cluster. Thus, we propose a simple k-medoids algorithm.

2.2. Proposed K-Medoids Algorithm

Even though SFKM can adopt any distances, this algorithm was originally simulated in pure numerical and categorical datasets. It performs well in a balanced orientation of a simulated numerical dataset. Unfortunately, when it comes to an unbalanced orientation dataset, the result is poor (Figure 1a). The dataset in Figure 1a has 800 objects with two variables [20]. The objects are assigned into five clusters, and the clusters have a spatially-unbalanced orientation in a two-dimensional space.

Thus, we develop a simple k-medoids (SKM) algorithm, which also applies medoid updating within its clusters in k-medoids setting. The term "simple" is borrowed from the SFKM algorithm for the initial medoids selection. What distinguishes the SFKM and SKM algorithms is that SKM considers the local optima and empty clusters. To minimize the local optima problem, several initializations have been used [21].

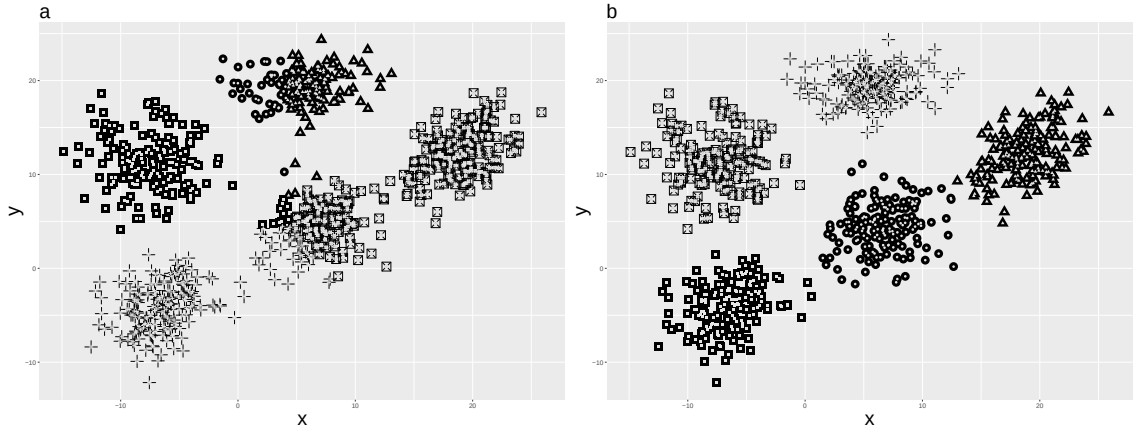

Figure 1. The simple and fast k-medoids (SFKM) (**a**) and simple k-medoids (SKM) (**b**) partitioning results.

The SKM algorithm is as follows:

1. Select a set of initial medoids $\mathcal{M}_k$, i.e., selecting k objects from X_n, randomly.
   The random initialization is preferred because the random initialization repeated several times is the best approach [21]. However, a restriction is applied to a single point of the $\mathcal{M}_k$ set. One of the $\mathcal{M}_k$ sets, which is randomly selected, is replaced by the most centrally-located object. It is argued that an optimal medoid is when $k = 1$ is the centrally-located object that minimizes the sum distance to the other objects. Then, the most centrally-located object can be selected based on the row/column sum of the distance matrix. It is given as:

$$a_i = \sum_{j=1}^{n} d_{ij} = \sum_{j=1}^{n} d_{ji}, \quad i = 1, 2, 3, \ldots, n,$$

   where a_i is the sum distance of the object i to the other objects. If vector a_i is ordered, the first $\mathcal{M}_k$ is selected by the smallest a_i, which is the most centrally-located object. Note that one/more medoids from the medoids set $\mathcal{M}_k$ can be non-unique objects, for example, two medoids have equal values in all their variables.
2. Assign the label/membership of each object $x_n \in X_n$, $l(x_n)$, to the closest medoid $\mathcal{M}_k$.
   When the medoid set of $\mathcal{M}_k$ is non-unique objects, an empty cluster can occur because the non-unique medoids and their closest objects will have the same membership label. To avoid an empty cluster, the non-unique medoids are restricted to preserve their label. While the closest

objects to the non-unique centroids can be labeled as any cluster membership in the k-means case [18], we regulate that the closest objects to the non-unique medoids are assigned/labeled as only one of the medoids, i.e., a single cluster membership. Hence, this guarantees a faster medoid updating step provided that the non-unique medoid set is not the most centrally-located object.

3. Update the new set of medoids $\mathcal{M}_k$, maintaining the clusters label $l(x_n)$ fixed.

$$m_p := \underset{p \in K}{argmin} \sum_{n:l(x_n)=m_p} d(x_n, m_p),$$
$$p = 1, 2, 3, \ldots, k$$

A medoid, m_p, is defined as an object that minimizes the sum distance of this object to the other objects within its cluster.

4. Calculate the sum of each cluster sum distance (E), i.e., the sum distance between objects and their medoids.

$$E = \sum_{p=1}^{k} \sum_{n:l(x_n)=m_p} d(x_n, m_p).$$

5. Repeat Steps 2–4 until E is equal to the previous E, or the set of medoids $\mathcal{M}_k$ does not change, or a pre-determined number of iterations is attained.
6. Repeat Steps 1–5 s times to attain multiple random seeding of the initial medoids.
   Note that the centrally-located object always becomes one of the initial medoids. Among the cluster medoids from the multiple random seeding s, the cluster medoid set that has a minimum value of E is selected as the final cluster medoids.
7. Assign the membership of each object to the final medoids.

The SKM algorithm can be applied in the aforementioned dataset, which yields a better result than the SFKM algorithm (Figure 1b).

2.3. Proposed Distance Method

As a distance-based algorithm, the SKM algorithm input requires a distance function. Compared to distance measures in either numerical or categorical variables, there are limited alternatives for mixed variable distance. Although the most common distance measure for mixed variable data is the Gower distance [9], a generalized distance function (GDF) to increase the variation of the mixed variable choices is developed.

The GDF consists of combinations of weights and distances. It sets particular weights in a linear combination of numerical, binary, and categorical distances. The distance between two objects in the GDF form can be computed by:

$$d_{ij} = \left(\alpha \sum_{r=1}^{p_n} \delta_n(x_{ir}, x_{jr}) + \beta \sum_{t=1}^{p_b} \delta_b(x_{it}, x_{jt}) + \gamma \sum_{s=1}^{p_c} \delta_c(x_{is}, x_{js}) \right)^{\omega}, \quad (1)$$

where ω is the weight for the whole function, α, β, and γ are the weights for the numerical, binary, and categorical variables, respectively, p_n is the number of numerical variables, p_b is the number of binary variables, p_c is the number of categorical variables, δ_n, δ_b, and δ_c are the numerical, binary, and categorical distances, respectively, and $\delta_n(x_{ir}, x_{jr})$ is the numerical distance between object i and object j in the variable r.

α, β, and γ are constants implying that each type of all inclusion variables has its own contribution, and each individual variable in the same type equally contributes to the distance computation. When the relative influences of individual variables differ, which can be based on the user domain

knowledge or intuition, the constants can be transformed into a vector. This transformation, however, results in a subspace clustering domain [10], which is out of the scope of this article.

By using the GDF, we can reformulate the Gower distance, which is a special case of the GDF. The Gower similarity [9] between object i and object j is normally calculated as:

$$s_{ij} = \frac{\sum_{l=1}^{p} \omega_{ijl} s_{ijl}}{\sum_{l=1}^{p} \omega_{ijl}}, \tag{2}$$

$$\begin{aligned} s_{ijl} &= 1 - \frac{|x_{il} - x_{jl}|}{R_l}, \text{ or} \\ s_{ijl} &\in \{0, 1\}, \end{aligned}$$

where ω_{ijl} is the j^{th} weight of object i in the variable l, x_{il} is the value of the object i on the l^{th} (numerical) variable, R_l is the range of the l^{th} (numerical) variable, and $s_{ijl} = 1$ if the objects i and j are similar in the l^{th} (binary or categorical) variable and zero otherwise. The Gower distance is then obtained by subtracting one from Equation (2), i.e., $d_{ij} = 1 - s_{ij}$. ω_{ijl} in the Gower distance regulates the contribution of each variable. ω_{ijl} is equal to zero if there is a missing value [22]. When some specific variables are more relevant than others to cluster the objects, ω_{ijl} is unequal [10]. This occurs in the domain of subspace clustering. For a conventional clustering method, on the other hand, equal variable weights (ω_{ijl}) for all of the variables are applicable. Thus, the Gower distance for a non-missing value dataset and all variables being equally contributed can be formulated in a GDF form into:

$$d_{ij} = \frac{1}{p_n + p_b + p_c} \left(\sum_{r=1}^{p_n} \frac{1}{R_r} |x_{ir} - x_{jr}| + p_b \sum_{t=1}^{p_b} \delta_b(x_{it}, x_{jt}) + p_c \sum_{s=1}^{p_c} \delta_c(x_{is}, x_{js}) \right), \tag{3}$$

where R_r is the range of variable r, $\delta_b(x_{it}, x_{jt}) = \delta_c(x_{is}, x_{js}) = 0$ if object i and object j are similar and one otherwise, i.e., simple matching distance. With equally-contributing variables and without any missing value, Equations (2) and (3) are equal (see Appendix A). Equation (3) implies that the distance for the numerical variable is the Manhattan range weighted, while the distance for the binary and categorical variables is the simple matching.

The other available distances for mixed variable data, which are less popular than the Gower distance, can be transformed into a GDF form in a similar way. They are the Wishart [23], Podani [22], Huang [5], and Harikumar-PV [7] distances. The Wishart and Podani distances have similar properties to the Gower such that the same assumption applies. The Huang and Harikumar-PV distances, on the other hand, do not admit a missing value and have equally-contributing variables by definition. Table 1 shows the GDF formulation of these distances with respect to their distance combinations and weights. Then, other combinations of distances and weights are possible.

Table 1. Mixed variable distances in the GDF formulation.

GDF	ω	α	β	γ	$\delta_n(x_{ir}, x_{jr})$	$\delta_b(x_{it}, x_{jt})$	$\delta_c(x_{is}, x_{js})$
Gower	1	$\frac{1}{p_n+p_b+p_c}$	$\frac{p_b}{p_n+p_b+p_c}$	$\frac{p_c}{p_n+p_b+p_c}$	M rw	SM	SM
Wishart	$\frac{1}{2}$	$\frac{1}{p_n+p_b+p_c}$	$\frac{p_b}{p_n+p_b+p_c}$	$\frac{p_c}{p_n+p_b+p_c}$	SE vw	SM	SM
Podani	$\frac{1}{2}$	1	p_b	p_c	SE r^2w	SM	SM
Huang	1	1	$\frac{}{s_n}$	$\frac{}{s_n}$	SE	H	H
Harikumar-PV	1	1	1	1	M	H	CoC

p_n = number of numerical variables, p_b = number of binary variables, p_c = number of categorical variables, $\overline{s_n}$ = mean of the standard deviation of numerical variables, H = Hamming, M = Manhattan, SE = squared Euclidean, SM = simple matching, CoC = co-occurrence, rw = range weighted, r^2w = squared range weighted, vw = variance weighted.

3. Demonstration on Artificial and Real Datasets

To apply the proposed k-medoids algorithm (SKM) and distance (GDF), simulated datasets with specified classes ("true clusters") are generated. The authors of [24] generated artificial data with only numerical variables in arbitrary diameters, shapes, and orientations. In order to obtain a binary/categorical variable for the simulation data, a numerical variable is categorized via a quantile categorization. The categorization of a numerical variable into a binary/categorical variable is obtained by dividing the numerical variable into c classes with the $100/c$% quantile as the cut point. Thus, a binary variable is achieved when the median of the numerical variable is applied as a cutoff point.

The artificial data have a degree of separation parameter [25] that has to be pre-specified. The degree separation values range from −1–1, where one indicates the best-separated clusters. Due to the focus of the study on the different settings of the mixed variable datasets, the degree separation is fixed arbitrarily to 0.2, and all variables are important variables. The mixed variable datasets have four different settings. The variations are in the variables' proportion, number of clusters, number of variables, and number of objects.

Each setting of the artificial datasets is partitioned by applying the GDFs (Table 1) as the input. Due to the possibility of other combinations of distances and weights in the GDF, two distances derived from the GDF are added, namely the Esimma and Marweco GDFs (Table 2). While the former combines the Euclidean and simple matching distances, the latter sums the Manhattan range weighted distance with the co-occurrence distance. Both GDFs set all of the weights to one.

Table 2. Two examples of the mixed variable distances derived from the generalized distance function (GDF).

GDF	ω	α	β	γ	$\delta_n(x_{ir}, x_{jr})$	$\delta_b(x_{it}, x_{jt})$	$\delta_c(x_{is}, x_{js})$
Esimma	1	1	1	1	E	SM	SM
Marweco	1	1	1	1	M rw	CoC	CoC

E = Euclidean, CoC = co-occurrence, SM = simple matching, M = Manhattan, rw = range weighted.

Then, the k-medoids algorithms, which is compared in this article, are the PAM, the most popular medoid-based algorithm, SFKM, the evaluated algorithm, and SKM ($s = 20$), the proposed algorithm. To evaluate the algorithms' performance, the Rand indices are calculated and averaged via a subsetting strategy [26] in 50 replications of the seven GDFs (Tables 1 and 2). The Rand index is given by:

$$ri = \frac{a+b}{{}^{n}C_2},$$

where a and b are the number of pairs of objects in the data. If both object pairs are in the same cluster under the "true cluster" and are also in the same cluster under the particular algorithm, it belongs to a. Meanwhile, b indicates that both object pairs are in a different cluster in both the "true cluster" and particular algorithm.

We also used six real datasets without any missing value from the UCI machine learning repository [27]. These data are the Iris, Wine, Soybean, Vote, Zoo, and Credit approval datasets (Table 3). They were selected due to being commonly used to compare clustering algorithms. Each dataset was analyzed using the proposed SKM algorithm that was applied in the GDFs (Tables 1 and 2). The performance comparison among GDF variations was based on the clustering accuracy index where perfect clustering had a value of one, and failure of clustering was zero.

To produce the k-medoids algorithms and GDF variations, the analyses were run in the R software [28] applying several packages. The main package was the kmed package [20], which applied the SFKM algorithm and GDFs, while the PAM algorithm was applied in the cluster package [29]. The SKM algorithm, on the other hand, was written in an R code and is available in the Supplementary Material. The clusterGeneration [30] was also applied to generate the artificial data; the flexclust

package [31] had the Rand index function; and the clue package [32] were able to calculate the clustering accuracy. Last, the ggplot2 [33] and gridExtra [34] packages helped the visualization results. With an Intel i3 4GB RAM, all codes required about 90 min to produce the results.

Table 3. Details of the real datasets.

Data Set	n	p_n	p_b	p_c	k
Iris	150	4	0	0	3
Wine	178	13	0	0	3
Soybean	47	0	0	35	4
Vote	435	0	0	16	2
Zoo	101	1	15	0	7
Credit approval	653	6	0	9	2

n = number of objects, p_n, p_b, p_c = number of numerical, binary, and categorical variables, respectively, k = number of true clusters.

3.1. Data Simulation

3.1.1. Different Variable Proportions

The artificial data had four cases with different variable proportions. The variations of the variable proportions included mixed variable datasets with numerical, binary, and categorical variable domination. In any variable domination, the SFKM had the lowest Rand index and was significantly different from PAM and SKM (Table 4). When the variable domination was categorical variables with the number of variables equal to three in the Huang GDF, the Rand index of the SFKM algorithm (0.70) was the highest compared to the other two algorithms. However, in this variable domination, high Rand indices were achieved in the Podani GDF (0.78) in both the PAM and SKM algorithms where the SFKM algorithm had only 0.71 of the Rand index.

Table 4. Rand indices of the different proportions of variables.

Domination	Algorithm	Gower	Wishart	Podani	Huang	Harikumar	Esimma	Marweco
Numerical	PAM	0.74	0.82	0.73	0.84	0.83	0.85	0.82
	SFKM	0.72 [a]	0.73 [a]	0.71 [a]	0.74 [a]	0.71 [a]	0.76 [a]	0.71 [a]
	SKM	0.74	0.80	0.73	0.84	0.83	0.85	0.82
Binary	PAM	0.78	0.78	0.77	0.77	0.76	0.73	0.74
	SFKM	0.74 [a]	0.73 [a]	0.73 [a]	0.74 [a]	0.72 [a]	0.73 [c]	0.73 [c]
	SKM	0.78	0.77	0.78	0.77	0.75	0.74	0.75
Categorical $c = 3$	PAM	0.78	0.76	0.78	0.69	0.68	0.67	0.74
	SFKM	0.71 [a]	0.72 [a]	0.71 [a]	0.70 [a]	0.68	0.67	0.70 [a]
	SKM	0.77	0.75	0.78	0.69	0.68	0.67	0.74
Categorical $c = 5$	PAM	0.77	0.82	0.77	0.79	0.78	0.76	0.77
	SFKM	0.71 [a]	0.75 [a]	0.71 [a]	0.79	0.79	0.76	0.69 [a]
	SKM	0.76	0.81	0.77	0.79	0.78	0.76	0.77

Significantly different ($\alpha = 0.05$) from: [a] PAM and SKM, [c] SKM.

3.1.2. Different Number of Clusters

The artificial data generated four different numbers of clusters. The numbers of clusters were 3, 4, 8, and 10. In any number of clusters, the SFKM algorithm always gave the smallest Rand index and was different from the other two algorithms (Table 5). When the number of clusters was small, i.e., $k = 3$, and the GDF was Marweco, the Rand index of the SFKM was the highest (0.84) compared to other GDFs. Both PAM and SKM, however, had a higher Rand index (0.88), and it was statistically different from that of the SFKM. Meanwhile, if the number of clusters was large ($k = 10$), the highest Rand index (0.91) of both the PAM and SKM was the Huang GDF. Although the Rand index of the

SFKM was slightly lower than the other two, i.e., 0.90, it was statistically different. The result showed that SFKM was the most underperforming algorithm.

Table 5. Rand indices of the different numbers of clusters.

k	Algorithm	Gower	Wishart	Podani	Huang	Harikumar	Esimma	Marweco
3	PAM	0.83	0.87	0.81	0.81	0.83	0.84	0.88
	SFKM	0.79 [a]	0.83 [a]	0.76 [a]	0.79 [a]	0.81	0.81 [a]	0.84 [a]
	SKM	0.83	0.87	0.81	0.79	0.81	0.84	0.88
4	PAM	0.76	0.88	0.74	0.89	0.90	0.90	0.84
	SFKM	0.72 [a]	0.84 [a]	0.73	0.86 [a]	0.86 [a]	0.88 [a]	0.80 [a]
	SKM	0.77	0.88	0.74	0.89	0.90	0.90	0.84
8	PAM	0.82	0.91	0.81	0.91	0.90	0.91	0.88
	SFKM	0.80 [a]	0.87 [a]	0.80 [a]	0.89 [a]	0.87 [a]	0.89 [a]	0.84 [a]
	SKM	0.81	0.90	0.81	0.91	0.90	0.90	0.87
10	PAM	0.86	0.92	0.85	0.91	0.91	0.91	0.89
	SFKM	0.84 [a]	0.88 [a]	0.85 [a]	0.90 [a]	0.88 [a]	0.90 [a]	0.85 [a]
	SKM	0.85	0.90	0.85	0.91	0.90	0.91	0.87

Significantly different ($\alpha = 0.05$) from: [a] PAM and SKM.

3.1.3. Different Numbers of Variables

The artificial data simulated a different number of mixed variables. The numbers of the mixed variables were 6, 8, 10, and 14. In any number of mixed variables, the SFKM algorithm showed underperforming results (Table 6). When the number of mixed variables was six and the GDFs were Huang and Harikumar, the Rand index of the SFKM algorithm (0.70) was the highest compared to the other two algorithms. Nevertheless, high Rand indices were attained by the Gower GDF (0.85) in both the PAM and SKM algorithms, where the SFKM algorithm had only 0.79 of the Rand index.

Table 6. Rand indices of the different numbers of variables.

p	Algorithm	Gower	Wishart	Podani	Huang	Harikumar	Esimma	Marweco
6	PAM	0.85	0.77	0.83	0.69	0.69	0.69	0.79
	SFKM	0.79 [a]	0.75 [b]	0.81 [a]	0.70 [b]	0.70 [c]	0.70	0.77 [c]
	SKM	0.84	0.76	0.84	0.69	0.69	0.69	0.79
8	PAM	0.80	0.93	0.79	0.90	0.90	0.92	0.86
	SFKM	0.74 [a]	0.86 [a]	0.75 [a]	0.88 [a]	0.84 [a]	0.88 [a]	0.79 [a]
	SKM	0.80	0.93	0.79	0.91	0.90	0.92	0.86
10	PAM	0.74	0.77	0.73	0.76	0.76	0.76	0.76
	SFKM	0.71 [a]	0.73 [a]	0.71 [a]	0.74 [a]	0.73 [a]	0.74 [a]	0.70 [a]
	SKM	0.74	0.76	0.73	0.75	0.76	0.76	0.75
14	PAM	0.69	0.85	0.68	0.95	0.95	0.95	0.89
	SFKM	0.66 [a]	0.78 [a]	0.66 [a]	0.85 [a]	0.81 [a]	0.86 [a]	0.79 [a]
	SKM	0.69	0.85	0.67	0.95	0.95	0.95	0.89

Significantly different ($\alpha = 0.05$) from: [a] PAM and SKM, [b] PAM, [c] SKM.

3.1.4. Different Numbers of Objects

The artificial data had variations in the number of objects. The datasets had 100, 500, 1000, and 2000 objects. Overall, both PAM and SKM had better results than SFKM (Table 7). They shared similar high Rand indices ($\geq$0.95) when the number of objects was 2000 in the Huang, Harikumar-PV, and Esimma GDFs. However, although SFKM had a high index in the Harikumar-PV GDF, it was significantly different from both the PAM and SKM algorithms.

Table 7. Rand indices of the different numbers of objects.

n	Algorithm	Gower	Wishart	Podani	Huang	Harikumar	Esimma	Marweco
100	PAM	0.75	0.87	0.73	0.94	0.93	0.95	0.91
	SFKM	0.72 [b]	0.84 [a]	0.73	0.89 [a]	0.88 [a]	0.89 [a]	0.85 [a]
	SKM	0.74	0.88	0.73	0.94	0.93	0.94	0.91
500	PAM	0.76	0.93	0.75	0.91	0.90	0.92	0.92
	SFKM	0.74 [a]	0.88 [a]	0.74	0.88 [a]	0.87 [a]	0.90 [a]	0.89 [a]
	SKM	0.76	0.93	0.75	0.91	0.89	0.92	0.92
1000	PAM	0.75	0.88	0.75	0.86	0.89	0.89	0.85
	SFKM	0.73 [a]	0.81 [a]	0.74	0.86	0.86 [a]	0.87 [a]	0.84
	SKM	0.76	0.88	0.74	0.86	0.89	0.89	0.85
2000	PAM	0.77	0.96	0.77	0.98	0.98	0.98	0.95
	SFKM	0.76 [a]	0.91 [a]	0.76 [a]	0.97	0.95 [a]	0.97	0.92 [a]
	SKM	0.78	0.96	0.77	0.98	0.98	0.98	0.95

Significantly different ($\alpha = 0.05$) from: [a] PAM and SKM, [b] PAM, [c] SKM.

Tables 4–7 show that the proposed k-medoids (SKM) had better performance than the evaluated algorithm (SFKM). In all settings, the Rand indices between the "true cluster" and SKM algorithm were higher than those between the "true cluster" and SFKM algorithm. On the other hand, from the running time perspective, theoretically, the SKM algorithm required more time than the SFKM algorithm due to repetition sets vs. a single set of initial medoids. By applying s times the multiple random seeding, the SKM finished a partitioning algorithm in a maximum of s multiplied by the required time of the SFKM. A simulation of $k = 3$ and $k = 10$ with $s = 20$ in the SKM, however, had an insignificant gap from the SFKM when the number of clusters was large (Figure 2).

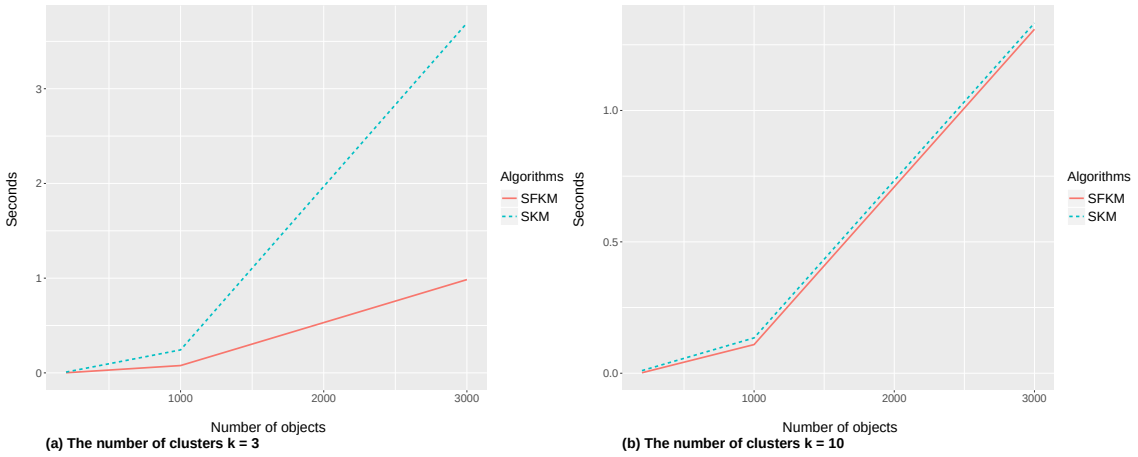

Figure 2. Benchmarking of the SKM and SFKM for $k = 3$ (**a**) and $k = 10$ (**b**).

Although the SFKM algorithm, which applied a specific formula for the initial medoid selection, gained speed, it performed badly with respect to the external validation, i.e., Rand index. With the current advancement in computation, in addition, the speed of the SKM algorithm could be increased. Due to the independent random seedings as many as s times, parallel computation was feasible to gain speed in the SKM. Hence, the SKM could be as fast as the SFKM when the number of cores was s. Although it was suggested to repeat the algorithm as many times as possible to avoid the local optima problem [17], with 20 seedings, the SKM algorithm yielded a similar result to the popular PAM algorithm. Then, the SKM and PAM algorithms were compared with respect to their efficiency.

The comparison took two different numbers of clusters ($k = 3$, 10) in small and large numbers of objects ($n = 200$, 1000, 3000). Each evaluation was averaged over 50 replications with seconds as the evaluation unit. The comparison also applied two different random number generators (*set.seed*).

Figures 3 and 4 show that PAM had a faster run time than SKM when the number of clusters was small ($k = 3$) and the number of objects was large ($n > 1000$). With large k and n ($k = 10, n > 1000$), on the other hand, SKM required less time compared to PAM. Thus, SKM and PAM were on par for small n and k.

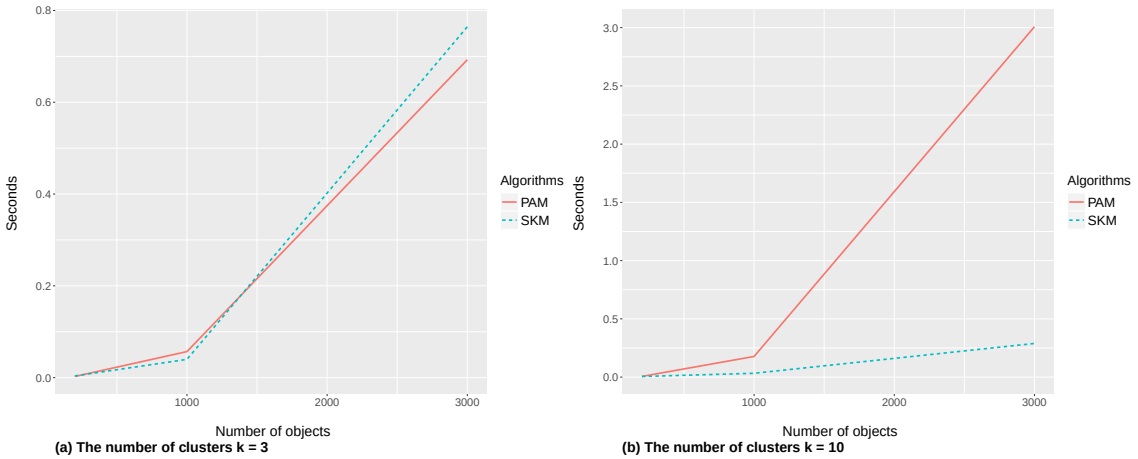

Figure 3. Benchmarking of PAM and SKM for $k = 3$ (**a**) and $k = 10$ (**b**) with *set.seed* (2018).

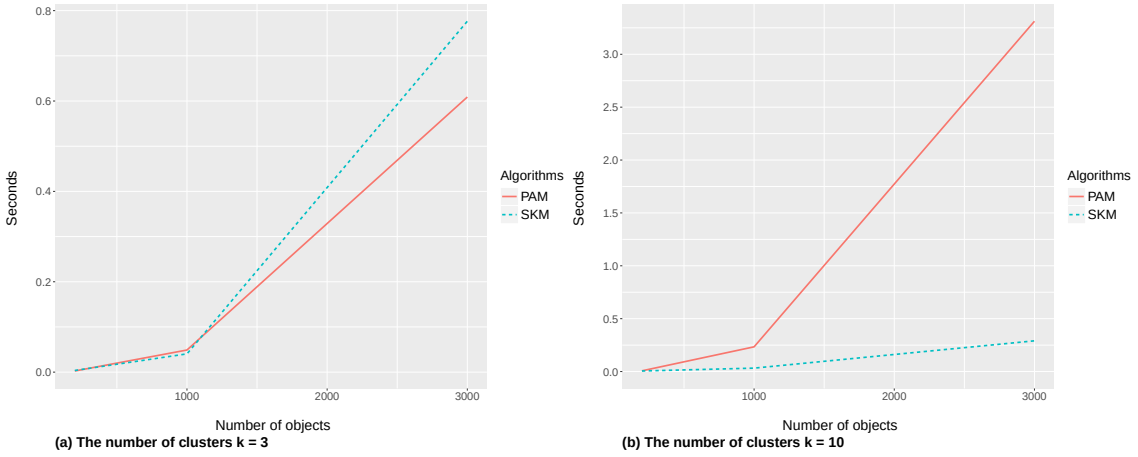

Figure 4. Benchmarking of PAM and SKM for $k = 3$ (**a**) and $k = 10$ (**b**) with *set.seed* (2019).

3.2. Real Datasets

3.2.1. Iris Data

The Iris dataset has no binary and categorical variables, albeit that it has only four numerical variables with 150 objects assigned to an equal size of three classes. We applied the seven GDFs and one additional Canberra GDF. The maximum clustering accuracy was 95.3% for the Canberra GDF. Comparing to the k-means [6] and k-prototype [13] for the mixed and categorical variable algorithms, the clustering accuracy of the SKM algorithm, which applied the Canberra GDF, was the highest (Table 8).

3.2.2. Wine Data

The wine dataset has 13 numerical variables with 178 objects assigned to three classes. The seven GDFs and Canberra GDF were applied. The highest accuracy index 92.7% was achieved when the Podani GDF was applied. If SFKM, k-harmonic means, and ranked k-medoids were applied to the wine data, the clustering accuracy indices gained only 70.4%, 68.5%, and 70.3%, respectively [19].

Table 8. Clustering accuracy (in percent) of the real datasets.

Dataset	Type	SKM	Other Algorithms
Iris	Numerical	95.3	94.7 [6] [a], 82.2 [13] [b]
Wine	Numerical	92.7	70.3 [19] [c]
Soybean	Categorical	100	91.0 [13] [b], 81.0 [16] [d]
Vote	Categorical	87.8	86.7 [6] [a]
Zoo	Mixed	82.2	78.7 [19] [c]
Credit approval	Mixed	82.7	77.9 [13] [b]

[a] k-means for mixed and categorical variables, [b] k-prototype for mixed and categorical variables, [c] rank k-medoids, [d] SFKM.

3.2.3. Soybean Data

The Soybean dataset only consists of categorical variables with 35 categorical variables and 47 objects attributed to four classes. Due to the absence of the numerical and binary variables, the seven GDFs were reduced to five GDFs. The two GDFs were the Esimma and Marweco GDFs that were equal to the Gower and Harikumar-PV, respectively. In the Soybean data, the Gower/ Esimma, Wishart, Harikumar-PV/ Marweco gained the maximum clustering accuracy, i.e., 100%. They out-performed both the k-prototype [13] and SFKM [16] algorithms.

3.2.4. Vote Data

The Vote dataset only consists of categorical variables with 16 categorical variables and 435 objects attributed to two classes. Harikumar-PV/ Marweco achieved 87.8% as the highest clustering accuracy. Compared to the k-prototype, hierarchical, and k-means for the mixed and categorical variables algorithms, they achieved 83.7%, 85.5%, and 86.7% clustering accuracy, respectively [6].

3.2.5. Zoo Data

The Zoo data have 101 objects assigned to seven classes. These data are mixed variables with one numerical and 15 binary variables. The seven GDFs were applied. The maximum clustering accuracy index was 82.2% for the Marweco GDF. Although the rank k-medoids [19] offered local optima way out through asymmetric distance, the clustering accuracy of the Zoo data was only 78.7% (Table 8).

3.2.6. Credit Approval Data

The Credit approval data have 653 objects (without NA) assigned to two classes. This dataset is mixed variables with six numerical and nine categorical variables. The seven GDFs were applied in the credit approval data. The maximum clustering accuracy index was 82.7% for the Marweco GDF. For this dataset, the author of [13] listed the clustering accuracy of other algorithms such as the k-prototype (56.2%), similarity-based agglomerative clustering (SBAC) (55.5%), KL-FCM-GM (Kullback–Leibler fuzzy c-means Gauss multinomial) (57.4%), and k-prototype for the mixed and categorical variables (77.9%) [13].

Table 8 shows that the SKM had better performances in all real datasets. Notice that the other algorithms in Table 8 may apply a different distance method. The SKM algorithm in the Iris dataset, for instance, applied the Canberra GDF that was compared to the k-means and k-prototype with the Euclidean distance. Although it appeared to be an unfair comparison due to the different GDF (distance) selections, our comparison emphasized the GDF's advantage in its combinations of the distances and weights.

The flexible combinations of the distances are an important feature of the GDF. While there are many alternatives for numerical, binary, and categorical distances, the mixed variable distances have few options. The more variations in the mixed variable distance are of the utmost importance due to the distance function exerting a major effect on a partitioning algorithm's result [35], i.e., different distances producing different optima. Hence, with the GDF's flexibility, various distance combinations

and weights can be simply applied. The Esimma and Marweco GDFs were the two GDF examples that were applied showing the GDF benefit.

Due to the linear combination of the numerical, binary, and categorical variables independently, another advantage of the GDF is that each distance class can be also canceled out when a class variable is excluded in a dataset. If a dataset only consists of numerical variables, for instance, the distance involved is only a numerical distance by eliminating the binary and categorical distances. The Canberra GDF in the Iris data, for example, emerged because the binary and categorical distances were omitted.

An adjustable weight to set a different contribution of an individual variable is the last advantage of the GDF. While the GDF had a scalar (constant) weight for α, β, and γ, it can be replaced by a vector of weights. The vector of weights implies that each variable has its own contribution to the distance computation, i.e., the variables do not equally contribute. Unequal contributions of the variables have been applied iteratively in the subspace clustering of the k-means algorithm by updated weights [36]. If a vector of weights is applied in the GDF, on the other hand, the contribution of the variables is calculated only once compared to as many as the iterations in the updated weights. Thus, the vector of weights in the GDF is preferred in a medoid-based algorithm setting than the updated weights.

4. Conclusions

A simple k-medoids (SKM) algorithm was proposed in this paper. To supply the distance as an input in this algorithm, a generalized distance function (GDF) was also developed. This function was adjustable to any type of data, and when the data were mixed variables, flexible combinations of the distances and weights were gained. Compared to the simple and fast k-medoids (SFKM) [16], which had a similar way in the within-cluster medoid updating, the SKM surpassed any mixed variable data case. The SKM performance was also similar to the PAM. With the s parameter as the number of initializations, a number of s equal to 20 was sufficient to produce a similar result to the PAM. When the numbers of clusters and objects were few, the run times of the SKM and PAM were similar. However, the SKM required less time for a large number of clusters and objects. The partitioning results of the six real datasets showed that the SKM had a better cluster accuracy index compared to other partitioning algorithms. Moreover, there is an R package, which is available from CRAN, called kmed, that implements both SFKM and GDF.

Supplementary Materials: The following are available online at: http://www.mdpi.com/1999-4893/12/9/177/s1. All Supplementary Materials are also available from the OSF (Open Science Framework) account of the corresponding author (http://www.doi.org/10.17605/OSF.IO/CNQTA).

Author Contributions: Conceptualization, W.B. and F.L.; methodology, W.B. and F.L.; programming and coding, W.B.; visualization, W.B.; writing, original draft preparation, W.B.; writing, review and editing, W.B. and F.L.; supervision, F.L.

Funding: This research was funded by Ditjen Sumberdaya IPTEK DIKTI (No. 138.44/E4.4/2015) and OeAD (ICM-2018-11915). The APC was funded by Institute of Statistics, University of Natural Resources and Life Sciences.

Acknowledgments: The authors would like to thank Ditjen Sumberdaya, the Ministry of Research, Technology, and Higher Education of Indonesia, Kemenristekdikti, and OeAD, Österreichischer Austauschdienst, for the continuous support via the Indonesia Austria Scholarship Program (IASP).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GDF	Generalized distance function
KM	K-medoids
PAM	Partitioning around medoids
SFKM	Simple and fast k-medoids
SKM	Simple k-medoids

Appendix A

The Gower distance of objects i and j is defined as $d_{ij} = 1 - s_{ij}$. Assuming no missing value and equally-contributing variables, i.e., $\omega_{ijl} = 1$, d_{ij} with a single variable of the numerical, binary, and categorical variables becomes:

$$\begin{aligned} d_{ij} &= 1 - s_{ij} \\ &= 1 - \frac{s_{ij}^{numerical} + s_{ij}^{binary} + s_{ij}^{categorical}}{1+1+1} \\ &= 1 - \frac{s_{ij}^{numerical} + s_{ij}^{binary} + s_{ij}^{categorical}}{3} \\ &= \frac{3 - (s_{ij}^{numerical} + s_{ij}^{binary} + s_{ij}^{categorical})}{3} \\ &= \frac{(1 - s_{ij}^{numerical}) + (1 - s_{ij}^{binary}) + (1 - s_{ij}^{categorical})}{3} \\ &= \frac{(1 - (1 - \frac{|x_i - x_j|}{R})) + d_{ij}^{binary} + d_{ij}^{categorical}}{3} \\ &= \frac{\frac{|x_i - x_j|}{R} + d_{ij}^{binary} + d_{ij}^{categorical}}{3}. \end{aligned}$$

When the numbers of numerical, binary, and categorical variables are p_n, p_b, and p_c, respectively, the Gower distance of objects i and j is:

$$\begin{aligned} d_{ij} &= \frac{\sum_{r=1}^{p_n} \frac{|x_{ir} - x_{jr}|}{R_r} + \sum_{t=1}^{p_b} d_{ij}^{binary} + \sum_{s=1}^{p_c} d_{ij}^{categorical}}{p_n + p_b + p_c} \\ &= \frac{\sum_{r=1}^{p_n} \frac{|x_{ir} - x_{jr}|}{R_r} + p_b \sum_{t=1}^{p_b} \frac{d_{ij}^{binary}}{p_b} + p_c \sum_{s=1}^{p_c} \frac{d_{ij}^{categorical}}{p_c}}{p_n + p_b + p_c} \\ &= \frac{1}{p_n + p_b + p_c} \left(\sum_{r=1}^{p_n} \frac{1}{R_r} |x_{ir} - x_{jr}| + p_b \sum_{t=1}^{p_b} \delta_b(x_{it}, x_{jt}) + p_c \sum_{s=1}^{p_c} \delta_c(x_{is}, x_{js}) \right), \end{aligned}$$

where both the $\delta_b(x_{it}, x_{jt})$ and $\delta_c(x_{is}, x_{js})$ are simple matching distances.

References

1. Gan, G.; Ma, C.; Wu, J. *Data Clustering, Theory, Algorithm, and Application*; The American Statistical Association; The Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2007.
2. Kaufman, L.; Rousseeuw, P.J. *Finding Groups in Data*; John Wiley and Sons Inc.: New York, NY, USA, 1990.
3. Hartigan, J.A.; Wong, M.A. A K-Means Clustering Algorithm. *J. R. Stat. Soc.* **1979**, *28*, 100–108.
4. Wu, X.; Kumar, V.; Quinlan, J.R.; Ghosh, J.; Yang, Q.; Motoda, H.; McLachlan, G.J.; Ng, A.; Liu, B.; Yu, P.S.; et al. Top 10 algorithms in data mining. *Knowl. Inf. Syst.* **2008**, *14*, 1–37. [CrossRef]
5. Huang, Z. Clustering large datasets with mixed numeric and categorical values. In *Proceedings of the First Pacific-Asia Conference on Knowledge Discovery and Data Mining*; Springer: Berlin, Germany, 1997; pp. 21–34.
6. Ahmad, A.; Dey, L. A K-mean clustering algorithm for mixed numeric and categorical data. *Data Knowl. Eng.* **2007**, *63*, 503–527. [CrossRef]
7. Harikumar, S.; Surya, P.V. K-Medoid Clustering for Heterogeneous DataSets. *Procedia Comput. Sci.* **2015**, *70*, 226–237. [CrossRef]
8. McCane, B.; Albert, M. Distance functions for categorical and mixed variables. *Pattern Recognit. Lett.* **2008**, *29*, 986–993. [CrossRef]

9. Gower, J.C. A General Coefficient of Similarity and Some of Its Properties. *Biometrics* **1971**, *27*, 857–871. [CrossRef]
10. Friedman, J.H.; Meulman, J.J. Clustering objects on subsets of attributes (with discussion). *J. R. Stat. Soc. Ser. B* **2004**, *66*, 815–849. [CrossRef]
11. Yin, J.; Tan, Z. Clustering Mixed Type Attributes in Large Dataset. In *ISPA 2005: Parallel and Distributed Processing Application*; Pan, Y., Chen, D., Guo, M., Cao, J., Dongarra, J., Eds.; Lecture Notes in Computer Science Volumn 3758; Springer: Berlin/Heidelberg, Germany, 2005; pp. 655–661.
12. Bushel, P.R.; Wolfinger, R.D.; Gibson, G. Simultaneous clustering of gene expression data with clinical chemistry and pathological evaluations reveals phenotypic prototypes. *BMC Syst. Biol.* **2007**, *1*, 1–20. [CrossRef]
13. Ji, J.; Bai, T.; Zhou, C.; Ma, C.; Wang, Z. An improved k-prototypes clustering algorithm for mixed numeric and categorical data. *Neurocomputing* **2013**, *120*, 590–596. [CrossRef]
14. Liu, S.H.; Shen, L.Z.; Huang, D.C. A three-stage framework for clustering mixed data. *WSEAS Trans. Syst.* **2016**, *15*, 1–10.
15. Reynolds, A.P.; Richards, G.; De La Iglesia, B.; Rayward-Smith, V.J. Clustering Rules: A Comparison of Partitioning and Hierarchical Clustering Algorithms. *J. Math. Model. Algorithms* **2006**, *5*, 475–504. [CrossRef]
16. Park H.; Jun, C. A simple and fast algorithm for K-medoids clustering. *Expert Syst. Appl.* **2009**, *36*, 3336–3341. [CrossRef]
17. Steinley, D. Local Optima in K-Means Clustering: What You Don't Know May Hurt You. *Psychol. Methods* **2003**, *8*, 294–304. [CrossRef]
18. Pakhira, M.K. A Modified k-means Algorithm to Avoid Empty Clusters. *Int. J. Recent Trends Eng.* **2009**, *1*, 221–226.
19. Zadegan, S.M.R.; Mirzaie, M.; Sadoughi, F. Ranked k-medoids: A fast and accurate rank-based partitioning algorithm for clustering large datasets. *Knowl.-Based Syst.* **2016**, *39*, 133–143. [CrossRef]
20. Budiaji, W. kmed: Distance-Based K-Medoids. R Package Version 0.3.0. 2019. Available online: http://CRAN.R-project.org/package=kmed (accessed on 15 June 2019).
21. Steinley, D.; Brusco, M. Initializing K-means Batch Clustering: A Critical Evaluation of Several Techniques. *J. Classif.* **2007**, *24*, 99–121. [CrossRef]
22. Podani, J. *Introduction to Exploration of Multivariate Biological Data*; Backhuys Publishers: Leiden, The Netherlands, 2000.
23. Wishart, D. K-Means Clustering with Outlier Detection, Mixed Variables and Missing Values. In *Exploratory Data Analysis in Empirical Research, Proceedings of the 25th Annual Conference of the Gesellschaft für Klassifikation e.V., University of Munich, 14–16 March 2001*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 216–226.
24. Qiu, W.; Joe, H. Generation of Random Clusters with Specified Degree of Separation. *J. Classif.* **2006**, *23*, 315–334. [CrossRef]
25. Qiu, W.; Joe, H. Separation Index and Partial Membership for Clustering. *Comput. Stat. Data Anal.* **2006**, *50*, 585–603. [CrossRef]
26. Hennig, C. Cluster-wise assessment of cluster stability. *Comput. Stat. Data Anal.* **2007**, *52*, 258–271. [CrossRef]
27. Lichman, M. *UCI Machine Learning Repository*; University of California: Irvine, CA, USA, 2013. Available online: http://archive.ics.uci.edu/ml (accessed on 17 July 2018).
28. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2015. Available online: https://www.R-project.org/ (accessed on 2 January 2017).
29. Maechler, M.; Rousseeuw, P.; Struyf, A.; Hubert, M.; Hornik, K. cluster: Cluster Analysis Basics and Extensions. R Package Version 2.0.6—For New Features, See the 'Changelog' File (in the Package Source). 2017. Available online: http://CRAN.R-project.org/package=cluster (accessed on 10 September 2018).
30. Qiu, W.; Joe, H. clusterGeneration: Random Cluster Generation (with Specified Degree of Separation). R Package Version 1.3.4. 2015. Available online: http://CRAN.R-project.org/package=clusterGeneration (accessed on 23 August 2018).
31. Leisch, L.; Dimitriadou, D.; Gruen, B. flexclust: Flexible Cluster Algorithms. R Package Version 1.4-0. 2018. Available online: http://CRAN.R-project.org/package=flexclust (accessed on 15 January 2019).
32. Hornik, K.; Böhm, W. clue: Cluster Ensembles. R Package Version 0.3-57. 2019. Available online: http://CRAN.R-project.org/package=clue (accessed on 7 May 2019).
33. Wickham, H. *ggplot2: Elegant Graphics for Data Analysis*; Springer: New York, NY, USA, 2016.

34. Auguie, B.; Antonov, A. gridExtra: Miscellaneous Functions for "Grid" Graphics. R Package Version 2.3. 2017. Available online: http://CRAN.R-project.org/package=gridExtra (accessed on 1 May 2018).
35. Leisch, F. A toolbox for K-centroids cluster analysis. *Comput. Stat. Data Anal.* **2006**, *51*, 526–544. [CrossRef]
36. Ahmad, A.; Dey, L. K-means type clustering algorithm for subspace clustering of mixed numeric and categorical datasets. *Pattern Recognit. Lett.* **2011**, *32*, 1062–1069. [CrossRef]