PRAGUE UNIVERSITY OF ECONOMICS AND BUSINESS

Acta Informatica Pragensia
2023, Volume 12, Issue 2, pp. 379–399
https://doi.org/10.18267/j.aip.223

Article Open Access

ck-means and *fck*-means: Two Deterministic Initialization Procedures for *k*-means Algorithm Using a Modified Crowding Distance

Abdesslem Layeb

LISIA Laboratory, NTIC Faculty, University of Constantine 2, Constantine, Algeria

Corresponding author: Abdesslem Layeb (abdesslem.layeb@univ-constantine2.dz)

Abstract

This paper presents two novel deterministic initialization procedures for *k*-means clustering based on a modified crowding distance. The procedures, named *ck*-means and *fck*-means, use more crowded points as initial centroids. Experimental studies on multiple datasets demonstrate that the proposed approach outperforms *k*-means and *k*-means++ in terms of clustering accuracy. The effectiveness of *ck*-means and *fck*-means is attributed to their ability to select better initial centroids based on the modified crowding distance. Overall, the proposed approach provides a promising alternative for improving *k*-means clustering.

Keywords

Clustering; *k*-means; *k*-means++; Initialization; Crowding distance; Heuristics.

Citation: Layeb, A. (2023). ck-means and fck-means: Two Deterministic Initialization Procedures for k-means Algorithm Using a Modified Crowding Distance. *Acta Informatica Pragensia*, 12(2), 379–399. https://doi.org/10.18267/j.aip.223

Academic Editor: Stanislav Vojir, Prague University of Economics and Business, Czech Republic

1 Introduction

k-means is a popular clustering algorithm that divides a dataset into *k* clusters based on similarity of data points. The algorithm is efficient and easy to implement but has some limitations, such as sensitivity to initial cluster centres, local optima and assumptions about cluster shapes. The algorithm proceeds by initializing *k* centroids, assigning data points to the nearest centroid, computing the mean of the data points in each cluster, and repeating until convergence (Ikotun et al., 2023; Hastie et al., 2009)

Despite its advantages, *k*-means clustering has some important considerations. The initial placement of centroids can affect the final clustering result, so multiple runs with different initializations may be necessary. The choice of "*k*" is also crucial and can be determined by domain knowledge or using techniques such as silhouette analysis or the elbow method. *k*-means may not always converge to the global optimal solution and is sensitive to outliers, which can lead to biased clustering results. Scaling of features is also necessary to avoid biased clustering results.

The initialization of *k*-means is a well-known challenge in the clustering community. In this paper, we address this challenge by proposing two novel initialization procedures for *k*-means: *ck*-means and *fck*-means. The *ck*-means procedure uses a modified crowding distance approach, inspired by the multi-objective optimization literature (Deb et al., 2002), to select the most representative initial centroids. The *fck*-means procedure, on the other hand, selects the furthest crowded points as initial centroids. Both procedures aim to provide a good deterministic initialization procedure that can improve the clustering results of *k*-means.

ck-means and *fck*-means are designed to address the limitations of standard initialization methods, such as the sensitivity to the initial placement of centroids and the assumption of spherical clusters with equal variance. The experimental results presented in Section 4 demonstrate the superiority of our proposed procedures over standard initialization methods, such as *k*-means++ and random initialization, in terms of clustering accuracy and stability.

Our proposed procedures have potential applications in various fields, such as image and text clustering, where the performance of *k*-means depends heavily on the initialization of centroids. Section 2 provides an overview of existing *k*-means initialization procedures. Section 3 presents a detailed description of the proposed initialization procedures and their implementation.

2 *k*-means algorithm

The *k*-means algorithm is a popular and widely used clustering algorithm that aims to divide a given dataset into *k* distinct clusters. It is an iterative algorithm that assigns each data point to the nearest centroid based on their distance and then updates the centroids based on the newly formed clusters. The algorithm searches for the minimum within-cluster sum of squares, also known as the inertia or distortion. The algorithm proceeds as follows:

1. Initialize *k* centroids randomly or using a more sophisticated initialization method, such as *k*-means++ (Arthur & Vassilvitskii, 2007).
2. Assign each data point to the nearest centroid based on a distance metric, such as Euclidean distance.
3. Recompute the centroids of the clusters by taking the mean of all data points assigned to each cluster. This step moves the centroids to the new centre of their respective clusters.
4. Repeat steps 2 and 3 until convergence, which is usually defined as the point when the positions of the centroids no longer change significantly.

The aim of *k*-means is to minimize the within-cluster sum of squares (WCSS):

$$WCSS = \sum_{k=1}^{K} \sum_{\substack{i=1 \\ (x_{i:} \in c_k)}}^{n_k} \sum_{j=1}^{p} (x_{ij} - m_{kj})^2 ,$$

where K is the number of clusters, n_k is the number of observations of the kk^{th} cluster and p is the number of features in a given dataset; x_{ij} is the value of the j^{th} feature of the i^{th} datapoint, $x_{i:}$ is the vector representing the i^{th} datapoint; m_{kj}: specifies the location of the k^{th} centroid.

The *k*-means algorithm is efficient and easy to implement but has some limitations, such as sensitivity to initialization, local optima and assumptions about cluster shapes. To address these limitations, several improvements have been proposed, such as advanced initialization methods, variations of the *k*-means algorithm, and integration with other clustering algorithms (Celebi et al., 2013; Liu et al., 2015; Hamerly & Elkan, 2003). One of the most critical challenges of *k*-means is the initialization of centroids, which can significantly affect the final clustering result. *k*-means++ is a popular initialization method that selects initial centroids that are far apart from each other and can improve the convergence rate of the algorithm and quality of the final clustering result (Arthur & Vassilvitskii, 2007).

Another critical limitation of *k*-means is that *k*-means works well with clusters having spherical shapes with equal variance, which may not hold for datasets with irregular shapes or different variances. To overcome this limitation, several variations of *k*-means have been proposed, such as fuzzy *k*-means, spectral clustering and hierarchical clustering, which can handle more complex data structures and cluster shapes (Xu & Wunsch, 2005; Fränti & Sieranoja, 2019).

In recent years, deep learning techniques, such as autoencoders and neural networks, have also been used to learn a better initialization for *k*-means. These methods can capture more complex relationships between data points and provide a more robust initialization procedure than traditional methods (Xie et al., 2016; Yang et al., 2017).

3 Initialization *k*-means methods

k-means is a popular clustering algorithm used in machine learning and data analysis. The algorithm requires the initialization of *k* centroids before it can partition the data points into *k* clusters. The quality of the initialization can significantly affect the performance and final clustering results of the algorithm. In this section, we will discuss several initialization methods for the *k*-means clustering algorithm.

One of the most popular initialization methods is *k*-means++, which was proposed by Arthur and Vassilvitskii in 2007. The *k*-means++ algorithm selects the first centroid randomly from the data points, and then selects the next centroids with a probability proportional to the distance from the data point to the nearest existing centroid. This approach tends to choose centroids that are well-spaced and can lead to better clustering results.

Another initialization method is the MaxMin method (Celebi et al., 2013), which selects the first centroid randomly from the data points, and then selects the subsequent centroids by choosing the data point with the maximum distance to the nearest previously selected centroid. This approach can also lead to well-spaced centroids, but it can be sensitive to outliers.

The PCA part method (Liu et al., 2015) starts with a single cluster containing all data points and completes the process in *k*-1 steps. In each step, it selects the cluster with the largest partial sum of squared Euclidean distances and divides it into two separate sub-clusters using a hyperplane that passes through the cluster centroid and is orthogonal to the direction of the principal eigenvector of the covariance matrix.

The global *k*-means (GKM) (Hamerly & Elkan, 2003) method starts with two centres and adds a centre each time by considering each data point as a candidate for the next centre. The data point that leads to the minimum value of the objective function, which is the sum of squared Euclidean distances to the closest centre, is selected as the next centre. The modified GKM (MGKM) method proposes a different way to minimize an auxiliary cluster function to select the starting point of the next centre. The fast MGKM (FMGKM) method exploits information gathered in previous iterations of the incremental algorithm to decrease memory usage and increase speed.

To tackle initialization problems of the *k*-means algorithm, the MinMax (Tzortzis et al., 2014) version of *k*-means changes the objective function and uses maximum intra-cluster variance (εmax) as a potential objective function to be minimized. The MinMax assigns a weight to each cluster, such that clusters with larger intra-cluster variance are allocated higher weights, and these weights are learned automatically. This approach is less affected by initialization and can discover high-quality solutions, even with bad initial centres.

Kumar & Kumar (2017) presented a kernel density-based method to compute the centroids for the *k*-means algorithm. The idea is to select an initial point from the denser region because they truly replicate the property of the complete dataset. Subsequently, the outliers are not selected as an initial seed value.

The tri-level k-means (TKM) (Yu et al., 2018) method starts with a coarse clustering of the data points into *m* clusters using *k*-means++, where *m* is smaller than *k*. Then, it selects *k* clusters from the *m* clusters based on their sizes and distances; and assigns the remaining data points to the nearest selected cluster. Finally, it applies *k*-means++ to each selected cluster to obtain finer sub-clusters. The advantage of TKM is that it can reduce the computational complexity and memory requirement of *k*-means by dividing the data into smaller subsets. The disadvantage is that it may lose some information in the coarse clustering step.

The bi-layer k-means (BKM) (Yu et al., 2018) method is another improved k-means algorithm that addresses the problem of sensitivity to the initial cluster centers. It works by clustering the data into two layers: rough and fine. The rough layer clustering is performed using a smaller number of clusters (m) than the desired number of clusters (k). The fine layer clustering is performed on the rough layer clusters using the desired number of clusters (k). The advantage of BKM is that it can preserve more information than TKM by applying *k*-means++ to each coarse cluster. The disadvantage is that it requires more computations and memory than TKM.

The entropy-based initialization (EBI) (Chowdhury et al., 2021) method starts with a random selection of *k* data points as the initial centroids. Then, it computes the entropy of each data point based on its distance to the centroids and assigns it to the cluster with the minimum entropy. Next, it updates the centroids by taking the mean of all the data points assigned to each cluster. It repeats these steps until no data point changes its cluster assignment or a maximum number of iterations is reached. The advantage of EBI is that it can handle clusters of different sizes and densities by using entropy as a measure of similarity. The disadvantage is that it may be sensitive to outliers.

4 Presentation of *ck*-means and *fck*-means

4.1 Crowding distance

Crowding distance is a popular multi-objective optimization technique that measures the density of non-dominated solutions within a particular region of the objective space. The primary purpose of this technique is to maintain diversity in the population of solutions, preventing them from clustering around a particular region.

We adapt the concept of crowding distance as a criterion for selecting the most representative centroids in the *k*-means algorithm. In this adaptation, each data point represents a solution, and each feature

corresponds to an objective function. The rationale behind this initialization procedure is to choose the densest (most crowded) points as the initial centroids for *k*-means clustering.

The crowding distance for *k*-means works by calculating the average side length of the rectangle formed by the neighbouring points for each point. The crowding distance is the sum of the average side lengths in all the features. In general, the points with the lowest crowding distances are considered the most representative and are preferred over other points as initial centroids. Finally, the crowding distance of the points with the lowest and highest feature values is assumed to be infinite.

To illustrate the concept of crowding distance, Figure 1 can be used. The concept can be visualized using a scatter plot of points in a two-dimensional feature space. Each data point x_i is represented by a point in the plot. The crowding distance is calculated for each point based on its neighbouring points. The neighbouring points can be defined using a distance metric, such as Euclidean distance, where the closest point is considered a neighbour. To compute the crowding distance of each point x_i of a given feature j, all the points are sorted in ascending order. The crowding distance of each point x_i is calculated by averaging the side lengths of the rectangles formed by the neighbouring points x_{i-1} and x_{i+1} adjacent to it.

$$d_j(x_i) = x_{(i+1)} - x_{(i-1)}$$

This value represents the crowding distance of the point x_i for the feature j. As can be seen from Algorithm 1, the points in the dense regions have small crowding distances, indicating that they are crowded, while those in the sparse regions have large crowding distances, indicating that they are more diverse. The pseudo-code of the crowding distance for *k*-means initialization is given in Algorithm 1.

By quantifying the crowding distance, we gain insights into the density of points within the feature space, thereby identifying crowded regions. This information aids in selecting the most representative points, which can be used as initial centroids in clustering algorithms such as *k*-means.

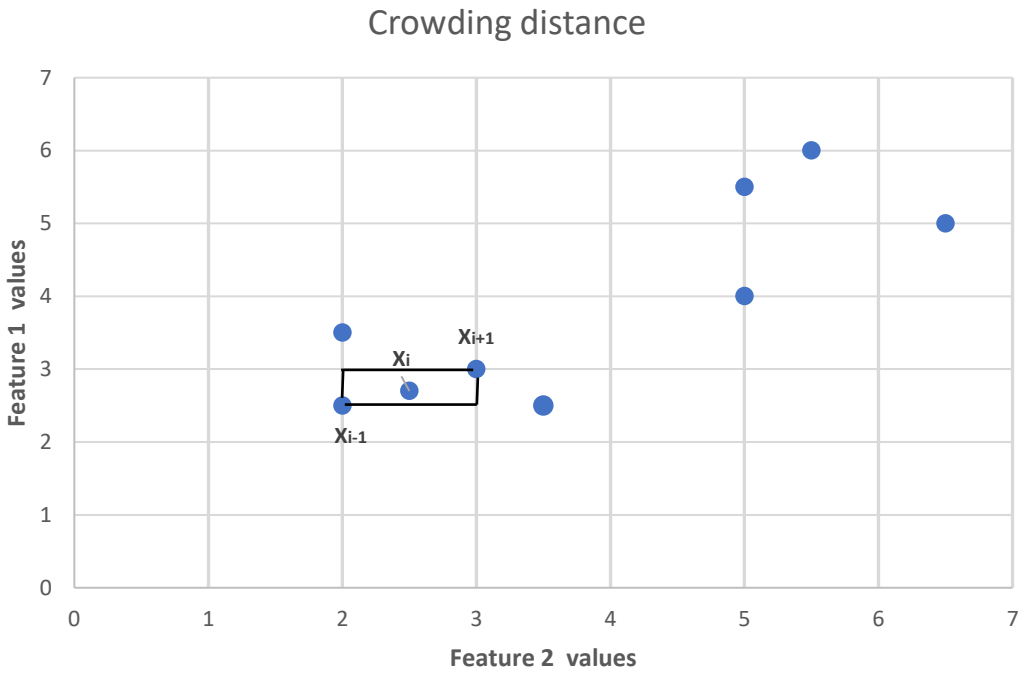

Figure 1. *Crowding distance.*

Input:

A set of data points S with M features

The maximum and minimum values of each feature

Output:

Crowding distances for each point in S

Algorithm:

For each point in S, initialize its crowding distance to 0.

For each feature j from 1 to M:

a. Sort the points in S according to their feature j values.
b. Set the crowding distance of the points with the lowest and highest feature values to infinity.
c. **For each** point x_i with feature value between the lowest and highest:
   - Calculate the distance between xi and its two neighbouring solutions in feature j dimension.
   - Add the average of the two distances to the crowding distance of x_i.

Return the crowding distances for each point in S.

Algorithm 1. *Crowding distance computation.*

Mathematically, crowding distance is computed as follows:

$$d_i(x_i) = \frac{1}{M}\sum_{j=1}^{M}\frac{f_j(x_{i+1}) - f_j(x_{i-1})}{f_j^{\max} - f_j^{\min}}$$

Where:

- $d_i(x_i)$ is the crowding distance of the point x_i;
- M is the number of objectives;
- $f_j(x_i)$ is the value of the feature j for the point x_i;
- $f_j^{\max}$ and $f_j^{\min}$ are the maximum and minimum values of the feature j in all the data points; and
- x_{i+1} and x_{i-1} are the neighbouring points of x_i in the feature j.
- The crowding distance of the extreme points is set as

$$d_1(x_1)=\infty,\ d_N(x_N)=\infty$$

Where:

- $d_1(x_1)$ is the crowding distance of the first sorting element;
- $d_N(x_N)$ is the crowding distance of the last sorting element; and
- N is the number of points in the dataset.

4.2 Modified crowding distances

The traditional method for computing crowding distances involves setting the distance values of the two extreme points to infinity, which can lead to biased results. To address this issue and obtain more accurate results, we modify the method in two ways. Firstly, we set the distance values of the extreme points to M times the maximum value of the point, where M is the number of features. Additionally, we introduce

two artificial points of data: the nadir point and the ideal point. The nadir point represents the minimum values across all features, while the ideal point represents the maximum values across all features. These artificial points provide reference values for the crowding distance calculation and help establish a more balanced and comprehensive assessment of point density within the dataset. Finally, the term $f_j^{\max} - f_j^{\min}$ is deleted to accelerate the distance computation. The modified crowding distance is computed as follows:

$$d_1(x_1) = M * Max(f^{\max})$$
$$d_{N+2}(x_{N+2}) = M * Max(f^{\max})$$

$$d_i(x_i) = \frac{1}{M}\sum_{j=1}^{M} f_j(x_{i+1}) - f_j(x_{i-1}) \;\; i = 2..N+1$$

4.3 Deterministic *k*-means initialization by crowding distance

In this work, we propose two new deterministic initialization procedures for clustering algorithms, based on the modified crowding. In the first one, called *ck*-means, the initial centroids are selected as follows:

1. Compute the crowding distance of all points.
2. Sort the distance in ascending order from the most crowded point to the least crowded point.
3. Select the first k points as centroids.

The motivation behind incorporating crowding distance into clustering algorithms is rooted in the observation that densely populated areas often correspond to clusters of similar data points. By identifying the most crowded points, which tend to lie at the core of these clusters, we gain insights into the underlying structure of the data and can effectively group similar points together. This concept is like the density-based clustering algorithm DBSCAN, which identifies clusters based on regions of high density in the data space. In both cases, the idea is to identify regions of the data space that are densely populated and use these regions as a basis for clustering or grouping similar solutions.

Utilizing crowding distance to identify the most crowded points offers several advantages. Firstly, it helps locate the core of each cluster, enabling a better understanding of the clustering structure of the data. Additionally, by ensuring that each cluster is well-separated from others, the crowding distance aids in improving the quality of the clustering results. This, in turn, enhances the interpretability and usefulness of the clusters for subsequent analysis tasks.

On the other hand, the *fck*-means algorithm is an enhancement of the *ck*-means algorithm, which uses the concept of furthest crowded points (FCPs) to initialize the centroids for the clustering process. The FCPs are selected based on a criterion that considers both the distance between points and their crowding distance. The main advantage of *fck*-means is that it ensures that the densest points in the feature space are far from each other, which can improve the quality of the resulting clusters and prevent the algorithm from converging to suboptimal solutions. The basic outline of the *fck*-means procedure is as follows:

1. Compute the crowding distance for each sample in the dataset.
2. Sort the points in descending order of their crowding distance.
3. Select the first point as the initial centroid.
4. For each subsequent point, compute the ratio of its distance to the current centroid and its crowding distance.
5. Sort the remaining points in descending order of this ratio.
6. Select the point with the highest ratio as the next centroid.
7. Repeat steps 4-6 until all centroids have been selected.

By using the FCPs as initial centroids, the *fck*-means algorithm is able to better capture the underlying structure of the data and produce more accurate and interpretable clustering results. Finally, we propose a random initialization version of *fck*-means, called *rck*-means (randomized *fck*-means). In this algorithm, the next centroid is selected randomly according to the probability of the distance-crowding ratio.

5 Experimental results

In this study, we utilized MATLAB Online 2023 to implement the suggested initialization procedures. We assessed their effectiveness by conducting evaluations on a total of 37 datasets, comprising both real and artificial datasets. These datasets encompass diverse data types and characteristics, with a comprehensive representation of 17 real datasets and 20 artificial ones. Table 1 presents extensive details about each dataset, including information such as data distribution, number of clusters and other pertinent attributes. Standardization is used to normalize the given dataset. To assess the effectiveness of the proposed procedures, we compared them with several popular initialization methods, such as *k*-means with random initialization, *k*-means++, MaxMinKmeans with MaxMin initialization, and MinMaxKmeans based on MinMax initialization. We used various evaluation metrics to measure the quality and robustness of the clustering results obtained by each initialization procedure for *k*-means clustering (Faridi et al., 2018; Maulik & Bandyopadhyay, 2002). These metrics include:

- Inertia score (IS): The inertia score measures the sum of squared distances between data points and their assigned centroids within a cluster. It quantifies how compact the clusters are and serves as a measure of clustering quality. A lower inertia score indicates better clustering, as it implies that data points within each cluster are closer to their respective centroids. The IS is calculated as the sum of squared distances between each data point x_i and its assigned centroid c_j within a cluster C_j:

$$IS = \sum_{j=1}^{k} \sum_{i=1}^{\|C_j\|} \|x_i - c_j\|^2$$

- Rand index score (RI): The Rand index score is a measure of similarity between two clusterings. It compares the number of true positives and true negatives between the two clusterings. The Rand index score ranges from 0 to 1, where a score of 1 indicates identical clusterings, and a score close to 0 indicates dissimilar clusterings. Given two clusterings X and Y, RI is computed as follows:

$$RI = \frac{a + b}{a + b + c + d}$$

  where:

  - a is the number of pairs that are in the same cluster in both X and Y (true positives),
  - b is the number of pairs that are in different clusters in both X and Y (true negatives),
  - c is the number of pairs that are in the same cluster in X but in different clusters in Y (false positives),
  - d is the number of pairs that are in different clusters in X but in the same cluster in Y (false negatives).

- Mutual information (MI): Mutual information measures the amount of information shared between two clusterings. It quantifies the similarity of the two clusterings by calculating the reduction in uncertainty in one clustering given the knowledge of the other clustering. A higher mutual information score indicates greater similarity between the clusters. Given two clusterings X and Y, MI is computed as follows:

$$MI = \sum_i \sum_i P(i,j) * \frac{\log\left(P(i,j)\right)}{P(i) * P(j)}$$

where:

- *P(i, j)* is the joint probability of data points being in the same cluster in both X and Y.
- *P(i)* is the probability of data points being in the same cluster in X,
- *P(j)* is the probability of data points being in the same cluster in Y.

- Silhouette index score (SI): The silhouette index score assesses the quality of clustering by measuring the distance between data points within clusters and between clusters. For each data point, it calculates a silhouette coefficient, which is a value between -1 and 1. A coefficient close to 1 indicates that the data point is well-matched to its own cluster and poorly matched to neighbouring clusters, indicating a good clustering. On the other hand, a coefficient close to -1 indicates that the data point may have been assigned to the wrong cluster. For each data point x_i, SI calculates the silhouette coefficient as follows:

$$SI(xi) = (b(i) - a(i)) / max(a(i), b(i))$$

where:

- *a(i)* is the average distance between x_i and all other data points in the same cluster; and
- *b(i)* is the minimum average distance between x_i and all data points in any other cluster.

The overall silhouette index score is the average of the silhouette coefficients for all data points.

$$SI = mean(SI(x_i))$$

- Davies Bouldin score (DB): The Davies Bouldin score measures the average similarity between each cluster and its most similar cluster while considering the scatter within the clusters. It penalizes clusters with high intra-cluster variance and encourages clusters that are well-separated. A lower Davies Bouldin score indicates better clustering, with values closer to 0 indicating more distinct and well-separated clusters. DB is calculated as follows:

$$DB = \frac{1}{k} * \sum \max\left[\frac{s(i) + s(j)}{d(c(i), c(j))}\right]$$

where:

- *k* is the number of clusters,
- *s(i)* is the scatter within the cluster *i*,
- *d(c(i), c(j))* is the distance between the centroids of the clusters *i* and *j*.

- Calinski Harabasz score (CH): The Calinski Harabasz score measures the ratio of between-cluster variance to within-cluster variance. It measures the separation between clusters by comparing the dispersion of data points within clusters to the dispersion between clusters. A higher Calinski Harabasz score indicates better clustering, with larger scores indicating more distinct and well-separated clusters. CH is calculated as follows:

$$CH = (Tr(B) / Tr(W)) * ((n - k) / (k - 1))$$

where:

- *Tr(B)* is the trace of the between-cluster scatter matrix,
- *Tr(W)* is the trace of the within-cluster scatter matrix,

- n is the total number of data points,
- k is the number of clusters.

The obtained results can be found in the Appendix. To facilitate the comparison of results for each initialization procedure, the metrics can be used. Lower values of IS and DB indicate better performance, while higher values of RI, MI, SI and CH indicate better performance. The mean results can be conveniently summarized in Tables A2 to A7, with the best results highlighted in boldface. Tables A2 to A7 provide a comprehensive overview of the scores for IS, RI, SI, MI, DB and CH, respectively.

Furthermore, the Friedman test can be employed to rank the different procedures and determine the overall best performer. Figures A1 to A6 illustrate the Friedman tests, comparing the scores for IS, RI, SI, MI, DB and CH, respectively. These figures aid in visualizing the relative performance and ranking of the different initialization procedures.

Based on the results presented in the following tables and Friedman tests, *ck*-means and *fck*-means are the top-performing initialization procedures for *k*-means clustering. They rank first in the inertia (Figure A1), RI (Figure A2), MI (Figure A4), and CH (Figure A6) metrics, which indicates that they perform well in terms of cluster quality and similarity to true clustering. On the other hand, MaxMinKmeans performs better in the SI (Figure A3) and DB (Figure A5) metrics, but is outperformed by *ck*-means and *fck*-means in the other metrics.

The effectiveness of *ck*-means and *fck*-means in improving *k*-means clustering is evident through their superiority over other *k*-means initialization procedures. The high scores achieved in metrics such as the Rand index (RI) and mutual information (MI) further affirm their ability to accurately identify clusters that closely align with true clustering.

6 Conclusion

This work proposed two novel initialization procedures for the *k*-means clustering algorithm called *ck*-means and *fck*-means. Both methods utilize a modified crowding distance approach to select the initial centroids. The experimental study showed that the proposed methods outperformed the standard initialization methods of *k*-means and *k*-means++ in terms of several metrics, including inertia, Rand index, mutual information and Calinski Harabasz. However, the MaxMinKmeans method performed better in the silhouette index and Davies Bouldin score metrics. The proposed methods can provide a more deterministic and effective initialization procedure for *k*-means, leading to better clustering results. Further research can investigate the application of the proposed methods in different clustering algorithms and explore their performance in real-world datasets.

Additional Information and Declarations

Conflict of Interests: The author declares no conflict of interest.

Author Contributions: The author confirms being the sole contributor of this work.

Data Availability: The data and code that support the findings of this study are openly available in Github repository at https://github.com/Layebuniv/fckmeans.

Appendix

Table A1. *Dataset details.*

	Dataset	Number of samples	Number of features	Number of clusters k
1	iris	150	4	3
2	ecoli	336	7	8
3	glass	214	9	2
4	balance	625	4	3
5	cancer	699	9	2
6	ovarian	216	100	2
7	thyroid	7200	21	3
8	sonar	208	60	2
9	chemical_test	498	8	40
10	ionosphere	351	34	2
11	data_heart	267	44	2
12	Zoo	101	16	7
13	SPECT	267	22	2
14	COIL20	1440	1024	20
15	semeion	1593	265	2
16	isolet	1559	617	26
17	house_test	506	13	46
18	dataset500_2_4	500	2	4
19	dataset500_2_5	500	2	5
20	dataset500_4_20	500	4	20
21	dataset1000_2_4	1000	2	4
22	dataset1000_2_5	1000	2	5
23	dataset1000_2_10	1000	2	10
24	dataset1000_4_3	1000	4	3
25	dataset1000_4_20	1000	4	20
26	dataset5000_2_4	5000	2	4
27	dataset5000_2_10	5000	2	10
28	dataset5000_4_3	5000	4	3
29	bdataset500_2_5	500	5	2
30	bdataset500_2_10	500	10	2
31	bdataset1000_2_4	1000	4	2
32	bdataset1000_2_10	1000	10	2
33	bdataset1000_4_3	1000	3	4
34	bdataset1000_4_20	1000	20	4
35	bdataset5000_2_4	5000	4	2
36	bdataset5000_2_10	5000	10	2
37	bdataset5000_4_3	5000	3	4

Table A2. *Inertia results.*

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
1	134.386	131.243	129.520	129.520	130.679	135.925	**129.407**
2	525.025	**393.760**	413.871	413.871	397.138	408.718	595.354
3	492.956	484.934	**471.241**	488.126	484.712	509.114	487.802
4	1007.386	1008.237	1063.160	1063.160	1007.436	1013.335	**1006.165**
5	1131.552	1131.454	**1128.707**	**1128.707**	1131.859	1132.077	1132.193
6	1326.453	1326.362	**1326.219**	1326.610	1326.430	1326.440	1326.378
7	25584.339	25142.727	24862.940	24862.940	24906.776	25517.035	**24694.141**
8	1441.403	1434.841	1429.051	**1428.408**	1437.318	1439.026	1435.718
9	523.113	508.142	519.097	508.049	**507.635**	517.971	521.898
10	1536.107	**1536.068**	1536.158	1536.158	1536.158	1612.035	1536.158
11	1412.978	1412.766	**1412.462**	**1412.462**	1412.782	1413.201	1413.196
12	223.514	219.744	**219.385**	219.614	224.191	224.989	225.736
13	**1106.897**	1107.026	1108.134	1108.134	1107.387	1107.608	1107.222
14	32267.446	27680.926	27595.535	28539.274	27750.364	30939.197	**27407.098**
15	25154.110	25145.762	**25127.698**	**25127.698**	25130.753	25150.728	25177.377
16	30171.132	**28214.945**	28385.241	28516.284	28231.818	28960.656	28253.812
17	621.886	549.936	684.460	582.240	**548.313**	558.125	642.557
18	209.369	205.525	**193.817**	**193.817**	209.509	199.700	216.965
19	236.268	237.418	228.487	**228.350**	233.817	233.850	238.297
20	189.739	170.236	253.506	**159.286**	169.860	160.484	265.006
21	458.980	462.823	436.035	436.035	447.496	439.830	**436.006**
22	424.380	410.340	413.042	**408.260**	410.408	410.391	410.223
23	234.124	216.877	272.051	**200.722**	219.350	212.997	301.863
24	561.528	581.104	**532.151**	**532.151**	571.304	561.532	551.746
25	413.970	376.098	380.238	**318.284**	364.895	341.827	493.730
26	2187.791	2116.649	**1974.367**	**1974.367**	2134.435	2187.791	2134.435
27	1129.651	1082.348	1339.973	**1033.059**	1089.522	1079.989	1178.448
28	3078.825	3224.392	**2933.258**	**2933.258**	3078.825	3661.087	**2933.258**
29	851.028	**850.980**	851.034	851.034	851.025	851.028	851.034
30	**1258.526**	**1258.526**	**1258.526**	**1258.526**	**1258.526**	**1258.526**	**1258.526**
31	1466.262	1466.387	1465.836	**1465.810**	1466.287	1466.721	1465.948
32	2687.197	2687.199	**2687.030**	2687.119	2687.249	2687.236	2687.331
33	1121.338	1120.071	1139.300	1117.453	1119.135	**1117.180**	1120.028
34	3651.794	3630.347	3578.537	**3578.522**	3652.550	3718.965	3712.702
35	8027.733	8027.773	8028.103	8028.103	8027.760	8027.910	**8027.685**
36	12950.388	12950.383	**12950.265**	12950.461	12950.409	12950.415	12950.383
37	5591.654	5591.402	**5590.934**	5593.180	5591.647	5591.330	5591.701

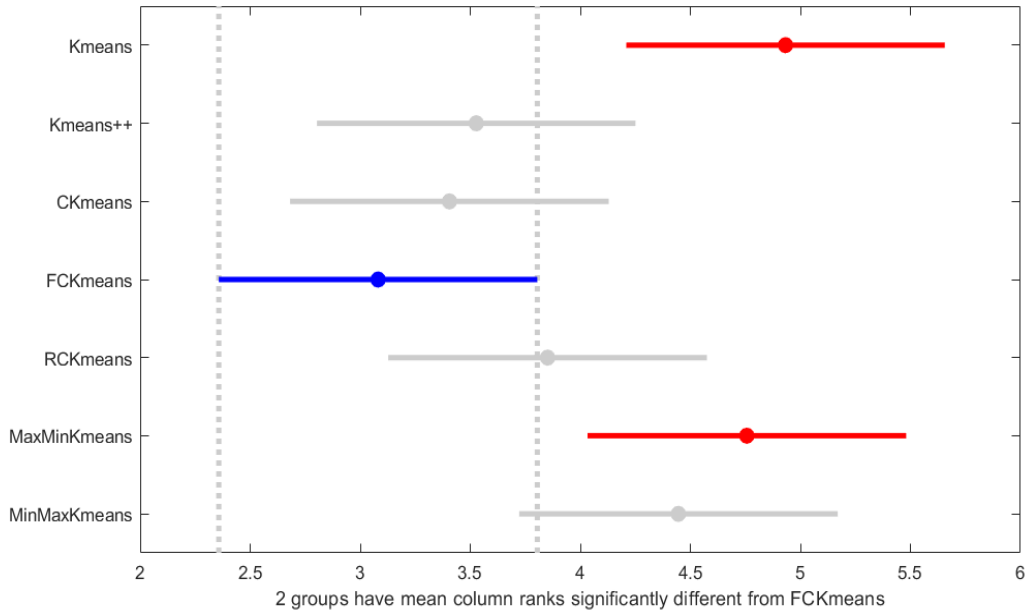

Figure A1. *Friedman test compares Inertia scores.*

Table A3. *RI score results.*

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
1	80.356	82.333	**84.313**	**84.313**	82.212	79.796	83.794
2	70.237	82.722	77.939	77.939	83.314	**85.208**	56.100
3	70.727	69.878	**83.744**	54.258	69.952	63.049	55.140
4	59.006	58.681	57.408	57.408	58.344	**59.722**	58.953
5	91.063	91.115	**92.562**	**92.562**	90.906	90.793	90.732
6	75.053	75.672	**76.645**	73.992	75.185	75.142	75.539
7	58.428	65.771	**75.235**	**75.235**	66.218	55.385	56.254
8	50.324	50.552	**51.101**	50.804	50.310	50.341	50.056
9	92.195	93.190	93.477	**93.601**	93.157	92.115	92.985
10	58.446	**58.470**	58.414	58.414	58.414	57.287	58.414
11	54.989	54.901	54.662	54.662	54.869	**55.133**	55.108
12	92.022	89.337	83.584	82.356	90.051	**94.725**	91.945
13	52.583	52.215	**53.164**	**53.164**	52.507	52.818	52.520
14	85.706	94.527	93.649	94.298	94.490	89.277	**94.681**
15	51.428	51.926	**52.921**	**52.921**	52.790	51.642	50.139
16	88.769	95.173	95.129	94.921	95.046	92.555	**95.180**
17	90.236	93.182	90.859	92.384	**93.279**	91.827	89.679
18	93.603	94.458	**96.987**	**96.987**	93.650	95.723	91.922
19	87.364	87.047	**90.308**	89.933	88.353	88.234	86.760
20	97.145	98.390	96.471	**99.598**	98.375	98.966	94.534
21	93.602	93.111	96.389	96.389	95.008	95.968	**96.431**
22	66.487	67.325	67.287	**67.579**	67.212	67.175	67.464
23	95.495	96.211	95.495	**97.741**	95.867	96.066	90.850

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
24	97.046	95.203	**99.867**	**99.867**	96.164	97.024	97.945
25	97.972	98.552	98.715	**99.980**	98.845	99.279	97.045
26	91.937	93.944	**97.959**	**97.959**	93.442	91.937	93.442
27	96.724	97.047	95.880	**97.846**	96.905	97.082	96.449
28	97.222	94.444	**100.000**	**100.000**	97.222	86.109	**100.000**
29	84.937	**85.117**	84.915	84.915	84.948	84.937	84.915
30	**90.481**	**90.481**	**90.481**	**90.481**	**90.481**	**90.481**	**90.481**
31	79.313	79.287	**79.934**	79.471	79.343	79.103	79.420
32	79.800	79.816	**80.089**	79.625	79.641	79.677	79.923
33	68.225	68.273	66.179	**68.471**	68.265	68.173	68.147
34	91.842	93.340	**97.455**	97.359	91.775	86.732	87.192
35	63.504	63.495	63.450	63.450	63.491	63.481	**63.522**
36	90.443	90.448	**90.534**	90.390	90.429	90.424	90.448
37	**70.554**	70.529	70.418	70.552	70.518	70.536	70.537

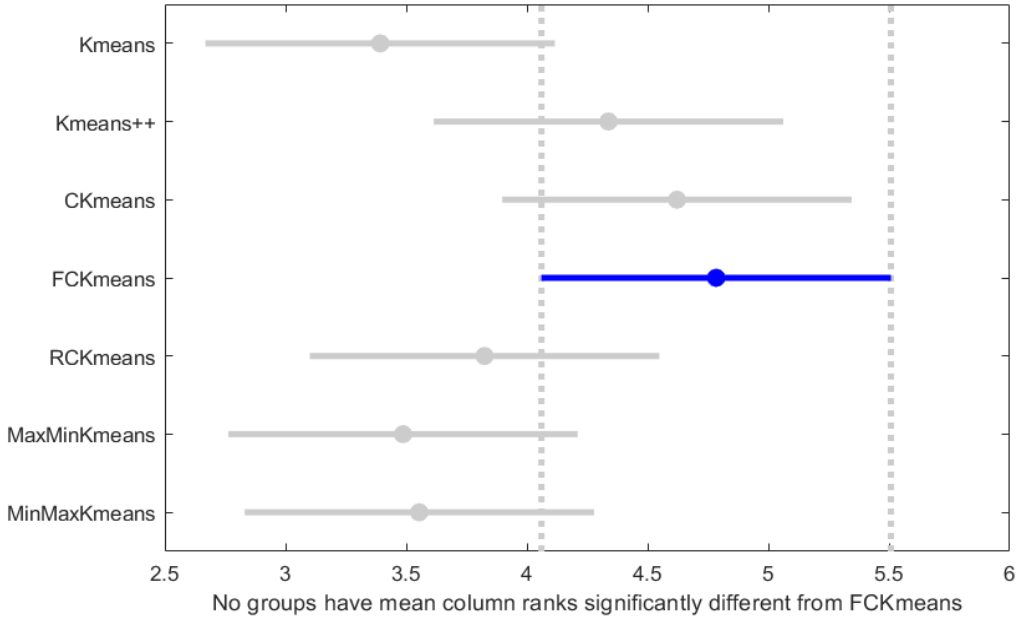

Figure A2. *Friedman test compares RI scores.*

Table A4. *SI score results.*

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
1	**0.653**	0.650	0.644	0.644	0.649	0.649	0.646
2	0.587	0.415	0.334	0.334	0.361	0.491	**0.601**
3	0.657	0.573	0.589	0.426	0.495	**0.721**	0.424
4	0.282	0.280	0.139	0.139	0.279	0.269	**0.287**
5	0.722	0.722	0.718	0.718	0.721	0.723	**0.723**
6	0.652	0.651	0.650	**0.654**	0.652	0.652	0.652
7	0.325	**0.371**	0.316	0.316	0.277	0.278	0.261
8	0.346	0.315	0.216	0.224	0.293	0.375	**0.392**

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
9	0.352	0.329	0.282	0.297	0.309	**0.379**	0.317
10	0.374	0.374	0.374	0.374	0.374	**0.434**	0.374
11	0.665	0.663	0.659	0.659	0.660	**0.668**	0.667
12	0.503	0.461	0.445	0.448	0.429	**0.557**	0.456
13	0.305	0.305	**0.322**	**0.322**	0.309	0.314	0.307
14	0.282	0.313	**0.314**	0.224	0.284	0.294	0.297
15	0.104	0.106	**0.110**	**0.110**	0.109	0.104	0.098
16	**0.167**	0.141	0.120	0.122	0.136	0.161	0.133
17	0.374	0.364	0.273	0.380	0.349	**0.421**	0.358
18	0.774	0.780	**0.805**	**0.805**	0.763	0.793	0.754
19	0.624	0.614	0.659	**0.664**	0.630	0.635	0.615
20	0.706	0.722	0.603	0.738	0.703	**0.753**	0.666
21	0.746	0.743	**0.769**	**0.769**	0.754	0.765	0.769
22	0.672	0.671	0.672	0.670	0.673	**0.674**	0.669
23	0.735	0.750	0.680	0.748	0.736	**0.758**	0.752
24	0.837	0.828	**0.852**	**0.852**	0.833	0.837	0.842
25	0.711	0.733	0.790	**0.837**	0.712	0.777	0.682
26	0.734	0.747	**0.774**	**0.774**	0.754	0.734	0.744
27	0.776	**0.783**	0.718	0.771	0.762	0.772	0.780
28	0.822	0.811	**0.833**	**0.833**	0.822	0.778	**0.833**
29	0.458	0.458	**0.458**	**0.458**	0.458	0.458	**0.458**
30	**0.419**	**0.419**	**0.419**	**0.419**	**0.419**	**0.419**	**0.419**
31	0.506	0.506	0.504	0.505	0.506	**0.507**	0.506
32	0.333	0.333	0.333	**0.334**	0.333	0.333	0.332
33	0.374	0.377	0.334	**0.383**	0.378	0.382	0.375
34	0.244	0.237	0.232	0.233	0.240	**0.265**	0.262
35	0.401	0.401	**0.402**	**0.402**	0.401	0.401	0.401
36	0.373	0.373	0.373	**0.373**	0.373	0.373	0.373
37	0.376	0.376	0.377	0.370	**0.377**	0.377	0.375

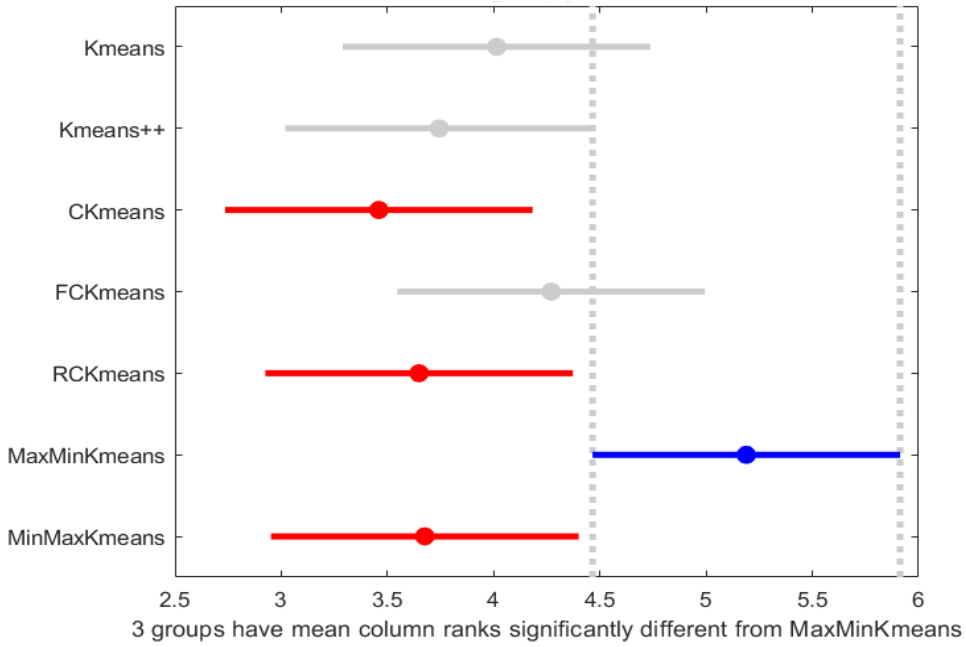

Figure A3. *Friedman test compares SI scores.*

Table A5. *MI score results.*

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
1	0.688	0.710	**0.726**	**0.726**	0.707	0.678	0.724
2	0.658	1.041	0.964	0.964	**1.044**	1.009	0.437
3	0.105	0.113	**0.255**	0.000	0.108	0.010	0.001
4	0.124	0.113	0.079	0.079	0.109	**0.141**	0.118
5	0.455	0.456	**0.482**	**0.482**	0.453	0.451	0.450
6	0.329	0.335	**0.345**	0.318	0.330	0.330	0.334
7	**0.012**	0.011	0.001	0.001	0.008	0.011	0.009
8	0.011	0.012	**0.014**	0.011	0.011	0.013	0.010
9	1.310	1.388	1.385	**1.412**	1.373	1.282	1.339
10	0.084	**0.085**	0.084	0.084	0.084	0.063	0.084
11	0.033	0.034	**0.035**	**0.035**	0.034	0.033	0.033
12	1.215	1.229	1.108	1.111	**1.229**	1.228	1.219
13	0.021	0.018	**0.024**	**0.024**	0.020	0.022	0.021
14	1.477	2.031	**2.124**	1.982	2.017	1.645	2.039
15	0.010	0.012	**0.018**	**0.018**	0.017	0.011	0.004
16	1.606	**2.111**	2.066	2.078	2.095	1.923	2.100
17	1.183	1.378	1.289	1.347	**1.388**	1.269	1.210
18	1.200	1.215	**1.263**	**1.263**	1.199	1.239	1.168
19	1.082	1.078	**1.151**	1.140	1.108	1.102	1.068
20	2.656	2.781	2.568	**2.912**	2.779	2.847	2.414
21	1.173	1.162	1.229	1.229	1.202	1.222	**1.231**
22	0.007	0.008	0.008	0.008	0.007	0.007	**0.008**
23	2.002	2.041	1.968	**2.110**	2.028	2.041	1.768

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
24	1.044	1.014	**1.092**	**1.092**	1.030	1.043	1.058
25	2.751	2.821	2.830	**2.986**	2.855	2.906	2.620
26	1.152	1.197	**1.287**	**1.287**	1.186	1.152	1.186
27	2.083	2.102	1.995	**2.136**	2.096	2.103	2.074
28	1.052	1.006	**1.099**	**1.099**	1.052	0.867	**1.099**
29	0.457	**0.459**	0.457	0.457	0.457	0.457	0.457
30	**0.526**	**0.526**	**0.526**	**0.526**	**0.526**	**0.526**	**0.526**
31	0.375	0.375	**0.380**	0.375	0.375	0.373	0.376
32	0.394	0.394	**0.397**	0.392	0.392	0.392	0.395
33	0.236	0.235	0.196	**0.236**	0.234	0.229	0.231
34	1.121	1.155	**1.252**	1.248	1.119	1.006	1.015
35	0.175	0.175	0.175	0.175	0.175	0.175	**0.175**
36	0.523	0.523	**0.524**	0.522	0.522	0.522	0.523
37	0.313	0.313	0.309	0.309	0.312	**0.314**	0.312

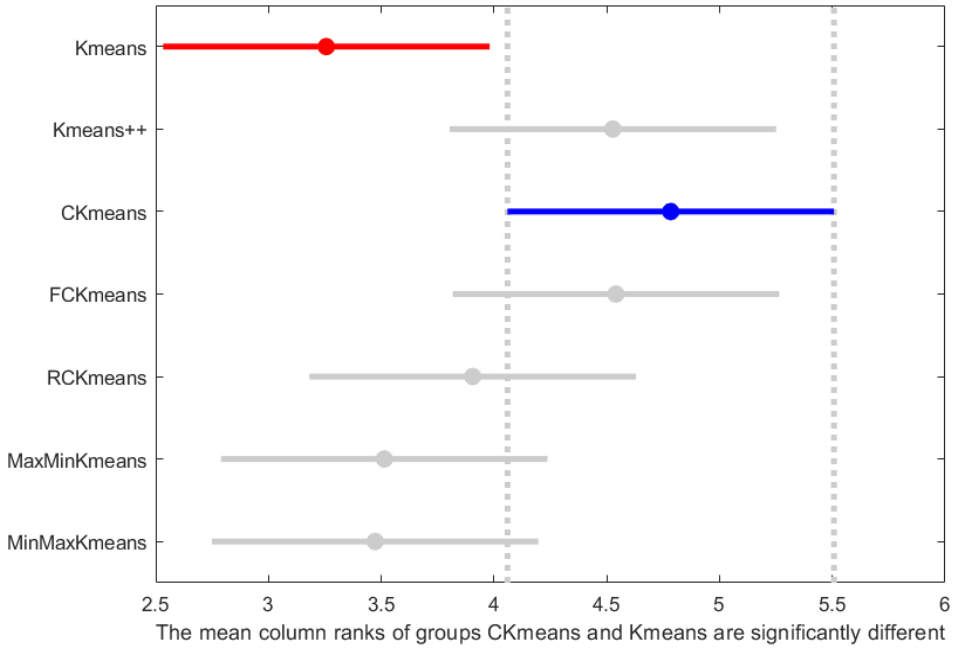

Figure A4. *Friedman test compares MI scores.*

Table A6. *DB score results.*

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
1	**0.802**	0.818	0.830	0.830	0.830	0.812	0.832
2	0.596	1.113	1.301	1.301	1.147	0.970	**0.591**
3	1.200	1.310	1.358	1.670	1.314	**0.839**	1.687
4	**1.708**	1.718	3.186	3.186	1.711	1.773	1.723
5	0.824	0.824	**0.823**	**0.823**	0.824	0.824	0.824
6	0.826	0.826	0.828	**0.824**	0.826	0.826	0.826

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
7	1.924	1.767	**1.219**	**1.219**	1.674	2.043	2.195
8	2.220	2.285	2.452	2.438	2.254	2.167	**2.086**
9	1.173	1.275	1.395	1.352	1.273	**1.105**	1.307
10	1.682	1.681	1.682	1.682	1.682	**1.355**	1.682
11	1.509	1.513	1.527	1.527	1.516	**1.501**	1.503
12	1.187	1.307	**1.127**	1.212	1.331	1.142	1.386
13	2.203	2.201	**2.174**	**2.174**	2.195	2.189	2.201
14	**1.525**	1.824	1.812	1.892	1.792	1.554	1.837
15	3.957	3.865	**3.671**	**3.671**	3.702	3.918	4.212
16	**1.554**	2.564	2.686	2.768	2.605	2.099	2.617
17	0.917	1.156	1.206	1.175	1.156	**0.901**	1.026
18	0.615	0.594	**0.534**	**0.534**	0.614	0.565	0.653
19	0.808	0.815	0.717	**0.703**	0.774	0.778	0.841
20	0.847	0.816	0.836	0.745	0.802	**0.690**	0.840
21	0.626	0.634	0.577	0.577	0.601	0.585	**0.577**
22	0.806	0.782	0.804	**0.754**	0.788	0.784	0.776
23	0.709	0.682	0.740	**0.638**	0.688	0.676	0.653
24	0.553	0.586	**0.502**	**0.502**	0.570	0.553	0.536
25	0.857	0.791	0.687	**0.533**	0.755	0.676	0.883
26	0.703	0.665	**0.589**	**0.589**	0.674	0.703	0.674
27	0.631	0.618	0.647	**0.602**	0.633	0.633	0.625
28	0.574	0.595	**0.553**	**0.553**	0.574	0.659	**0.553**
29	1.448	**1.448**	1.448	1.448	1.448	1.448	1.448
30	**1.600**	**1.600**	**1.600**	**1.600**	**1.600**	**1.600**	**1.600**
31	1.313	1.313	1.314	1.313	1.313	**1.313**	1.313
32	2.206	2.206	2.207	2.206	**2.205**	2.206	2.206
33	1.223	1.213	1.339	**1.193**	1.207	1.196	1.214
34	2.410	2.327	**2.032**	2.032	2.395	2.786	2.766
35	1.796	1.796	**1.794**	**1.794**	1.796	1.795	1.796
36	1.790	1.790	**1.790**	1.790	1.790	1.790	1.790
37	1.203	1.203	**1.202**	1.204	1.204	1.203	1.204

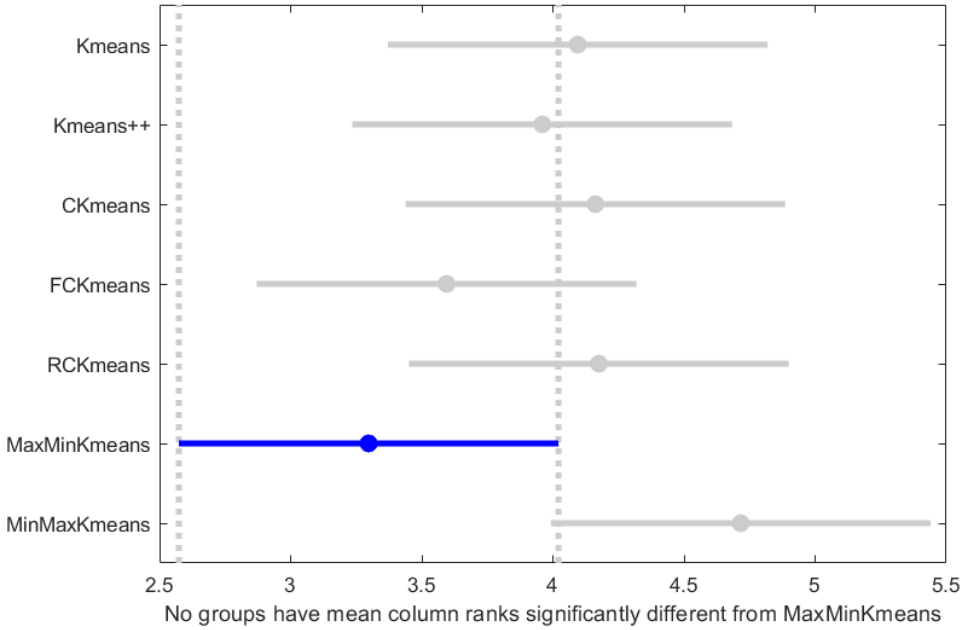

Figure A5. *Friedman test compares DB scores.*

Table A7. *CH score results.*

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
1	215.232	229.459	236.822	236.822	232.976	209.931	**237.783**
2	78.022	**140.424**	81.363	81.363	125.949	140.029	45.624
3	40.274	43.447	**55.084**	42.564	43.119	31.053	42.625
4	135.274	134.110	90.290	90.290	135.244	126.514	**136.738**
5	868.728	868.735	**868.892**	**868.892**	868.708	868.695	868.688
6	264.385	264.343	264.278	**264.455**	264.375	264.379	264.351
7	433.530	410.264	393.563	393.563	402.963	**441.704**	413.025
8	31.958	33.262	31.359	31.797	33.189	33.636	**35.302**
9	61.547	65.157	57.819	61.976	**65.409**	64.648	59.102
10	95.912	95.912	**95.913**	**95.913**	**95.913**	71.397	**95.913**
11	81.501	81.494	81.438	81.438	81.479	**81.529**	81.520
12	28.190	**28.883**	26.605	26.325	27.610	28.042	27.390
13	52.436	52.417	**52.755**	**52.755**	52.491	52.591	52.421
14	68.079	**103.920**	99.233	89.348	103.886	78.821	103.435
15	92.186	93.192	**95.400**	**95.400**	95.010	92.594	89.346
16	36.949	**51.072**	49.273	47.848	50.935	45.934	50.501
17	64.122	85.030	41.124	63.507	85.900	**85.959**	57.309
18	1378.217	1430.484	**1586.401**	**1586.401**	1378.211	1508.132	1274.183
19	754.581	743.836	**823.574**	822.754	775.681	775.791	737.793
20	571.652	717.030	251.804	811.489	719.513	**826.930**	248.715
21	2228.499	2192.027	2447.281	2447.281	2337.895	2410.828	**2447.293**
22	1969.872	2080.731	2065.777	**2096.985**	2081.555	2082.508	2083.163
23	3128.892	3593.734	1844.503	**4170.932**	3494.845	3685.366	1870.616

Test	Kmeans	Kmeans++	CKmeans	FCKmeans	RCKmeans	MaxMinKmeans	MinMaxKmeans
24	5072.050	4837.075	**5424.541**	**5424.541**	4954.583	5072.040	5189.527
25	947.049	1156.461	1086.584	**1741.277**	1245.405	1451.484	634.855
26	12543.112	13385.002	**15068.784**	**15068.784**	13174.530	12543.112	13174.530
27	16463.498	17929.471	9921.993	**19627.208**	17687.353	17994.394	15172.714
28	18876.527	17699.099	**20053.956**	**20053.956**	18876.527	14166.812	**20053.956**
29	206.453	206.449	**206.453**	**206.453**	206.453	206.453	**206.453**
30	**174.915**	**174.915**	**174.915**	**174.915**	**174.915**	**174.915**	**174.915**
31	491.319	491.322	**491.332**	491.311	491.322	491.317	491.309
32	186.960	186.956	**186.976**	186.973	186.958	186.959	186.892
33	315.727	317.722	293.203	**321.855**	318.531	321.251	317.658
34	131.038	134.335	144.332	**144.338**	131.168	119.451	120.395
35	1236.134	1236.137	**1236.138**	**1236.138**	1236.135	1236.136	1236.133
36	1405.291	1405.291	**1405.299**	1405.287	1405.290	1405.289	1405.291
37	1571.127	1571.497	**1571.763**	1568.493	1571.244	1571.654	1571.006

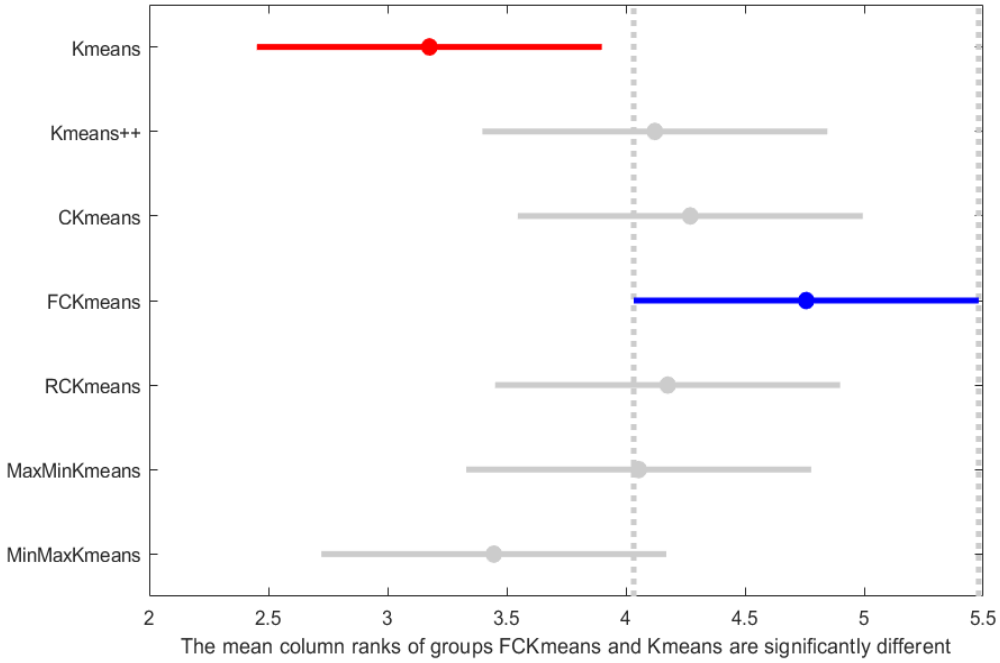

Figure A6. *Friedman test compares CH scores.*

References

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, (pp. 1027–1035). ACM.

Celebi, M. E., Kingravi, H. A., & Vela, P. A. (2013). A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems With Applications*, 40(1), 200–210. https://doi.org/10.1016/j.eswa.2012.07.021

Chowdhury, K., Chaudhuri, D., & Pal, A. K. (2021). An entropy-based initialization method of K-means clustering on the optimal number of clusters. *Neural Computing and Applications*, 33(12), 6965–6982. https://doi.org/10.1007/s00521-020-05471-9

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197. https://doi.org/10.1109/4235.996017

Faridi, H., Srinivasagopalan, S., & Verma, R. (2018). Performance evaluation of features and clustering algorithms for malware. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (pp. 13–22). IEEE. https://doi.org/10.1109/ICDMW.2018.00010

Fränti, P., & Sieranoja, S. (2019). How much can k-means be improved by using better initialization and repeats? *Pattern Recognition*, 93, 95–112. https://doi.org/10.1016/j.patcog.2019.04.014

Hamerly, G., & Elkan, C. (2003). Learning the k in k-means. In *Advances in neural information processing systems* (pp. 281–288). NeurIPS Proceedings. https://proceedings.neurips.cc/paper_files/paper/2003/file/234833147b97bb6aed53a8f4f1c7a7d8-Paper.pdf

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning. In *The Elements of Statistical Learning* (pp. 485–585). Springer. https://doi.org/10.1007/978-0-387-84858-7_14

Ikotun, A. M., Ezugwu, A. E., Abualigah, L., Abuhaija, B., & Jia, H. (2023). K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622, 178–210. https://doi.org/10.1016/j.ins.2022.11.139

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Kumar, A., & Kumar, S. (2017). Density based initialization method for K-Means clustering algorithm. *International Journal of Intelligent Systems and Applications*, 9(10), 40–48. https://doi.org/10.5815/ijisa.2017.10.05

Liu, H., Chen, F., Wu, Y., Xu, K., & Tian, D. (2015). Improved K-means Algorithm with the Pretreatment of PCA Dimension Reduction. *International Journal of Hybrid Information Technology*, 8(6), 195–204. https://doi.org/10.14257/ijhit.2015.8.6.19

Maulik, U., & Bandyopadhyay, S. (2002). Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12), 1650–1654. https://doi.org/10.1109/tpami.2002.1114856

Tzortzis, G., & Likas, A. (2014). The MinMax k-Means clustering algorithm. *Pattern Recognition*, 47(7), 2505–2516. https://doi.org/10.1016/j.patcog.2014.01.015

Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *Proceedings of The 33rd International Conference on Machine Learning*, (pp. 478–487). PMLR. https://proceedings.mlr.press/v48/xieb16.html

Xu, R., & Wunsch, D. C. (2005). Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16(3), 645–678. https://doi.org/10.1109/tnn.2005.845141

Yang, B., Fu, X., Sidiropoulos, N. D., & Hong, M. (2017). Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning*, (pp. 3861–3870). ACM.

Yu, S., Chu, S. W., Wang, C., Chan, Y., & Chang, T. (2018). Two improved k-means algorithms. *Applied Soft Computing*, 68, 747–755. https://doi.org/10.1016/j.asoc.2017.08.032

Editorial record: The article has been peer-reviewed. First submission received on 23 May 2023. Revision received on 7 August 2023. Accepted for publication on 3 September 2023. The editor in charge of coordinating the peer-review of this manuscript and approving it for publication was Stanislav Vojir.

Acta Informatica Pragensia is published by Prague University of Economics and Business, Czech Republic.

ISSN: 1805-4951