AC: A data generator for evaluation of clustering

Wenke Li [1] and Zhou Zhou [2]

[1]Fuwai hospital
[2]Affiliation not available

October 30, 2023

Abstract

Clustering has important applications in many fields. However, there are not enough benchmark datasets with rich characteristics for the development and evaluation of clustering algorithms, so the clustering performance cannot be truly evaluated. Neither real data nor manually synthetic data can solve this problem. We propose a new data generator, Artificial Cluster (AC, http://ac.fwgenetics.org), that can thoroughly customize the cluster characteristics that affect clustering, such as sample size, density, overlap and shape. The randomization of the default parameters enables AC to efficiently generate benchmark datasets with different characteristic combinations that can be used to evaluate the robustness of clustering algorithms. We evaluated nine popular clustering algorithms using an example benchmark dataset generated by AC. From the results, the advantages and disadvantages of these algorithms can be clearly seen. AC is expected to provide sufficient data support for clustering research.

Posted on 3 Feb 2020 — CC-BY-NC-SA 4 — https://doi.org/10.36227/techrxiv.19091330.v1 — e-Prints posted on TechRxiv are preliminary reports that are not peer reviewed. They should not b...

AC: A data generator for evaluation of clustering

Wenke Li [a], Zhou Zhou [a, *]

[a]State Key Laboratory of Cardiovascular Disease, Beijing Key Laboratory for Molecular Diagnostics of Cardiovascular Diseases, Center of Laboratory Medicine, Fuwai Hospital, National Center for Cardiovascular Diseases, Chinese Academy of Medical Sciences and Peking Union Medical College, North Lishi Road, Xicheng District No.167, Beijing 100037, China

ABSTRACT

Clustering has important applications in many fields. However, there are not enough benchmark datasets with rich characteristics for the development and evaluation of clustering algorithms, so the clustering performance cannot be truly evaluated. Neither real data nor manually synthetic data can solve this problem. We propose a new data generator, Artificial Cluster (AC, http://ac.fwgenetics.org), that can thoroughly customize the cluster characteristics that affect clustering, such as sample size, density, overlap and shape. The randomization of the default parameters enables AC to efficiently generate benchmark datasets with different characteristic combinations that can be used to evaluate the robustness of clustering algorithms. We evaluated nine popular clustering algorithms using an example benchmark dataset generated by AC. From the results, the advantages and disadvantages of these algorithms can be clearly seen. AC is expected to provide sufficient data support for clustering research.

Keywords: Benchmarking; Clustering; Data generator; Synthetic data; Performance evaluation

* Corresponding author.

E-mail addresses: wk1lian@126.com (Wenke Li), zhouzhou@fuwaihospital.org (Zhou Zhou)

1. Introduction

Clustering is one of the most important analysis methods in pattern recognition. Hundreds of clustering algorithms have been proposed [1]. However, these algorithms are usually sensitive to the distribution characteristics of clusters, such as density [2], [3], overlap [4], and shape [2], [5]. It is necessary to build benchmark datasets with rich characteristics for evaluating and improving clustering algorithms. There are many limitations when using real data as evaluation benchmarks, such as high cost, unevenness of quality, and difficulty controlling cluster distribution. In some studies of clustering algorithms, several datasets were synthesized manually [6]. However, the number and type of these datasets are too small to be used as a benchmark, which results in many clustering algorithms that cannot be effectively evaluated before they are used in practical applications. Therefore, automatic synthesis of datasets is the only way to solve this problem.

In fact, data generators have been widely used in big data benchmarking [7], but there is still not a powerful enough data generator for clustering because it is challenging to achieve flexible control over cluster characteristics in a synthetic dataset. Several studies have been reported (Table 1). generateData [8] generates two-dimensional data along a straight line. Scikit-learn [9] generates two-dimensional data along more shapes, including straight lines, double rings, S curves, and intersecting semicircles, among others. Elki [10] extended the synthetic data to a higher number of dimensions. The above methods lack control over the overlap between clusters. Three other studies [4], [11], [12] were able to control the overlap between clusters, but they could not simultaneously customize the cluster size, density, overlap and shape. Some solutions have been proposed for controlling cluster shape, such as combining data distributions [4], [12], introducing linear transformations [11], [12], [13], and built-in fixed shapes [13]. However, these methods either generate only convex clusters or a limited number of shapes and therefore cannot truly implement shape customization. In addition, randomness should be implemented along with control over cluster characteristics, as randomness is necessary for evaluating the robustness of a clustering algorithm and was often ignored by previous software. We propose a new data generator called Artificial Cluster (AC, http://ac.fwgenetics.org), which can thoroughly customize the cluster characteristics including sample size, density, overlap, and shape. Additionally, AC's parameters can be randomized so that a large number of datasets with similar characteristics can be generated quickly. By evaluating the results from the 9 clustering algorithms on a dataset generated by AC, we could see AC's potential to mine algorithm defects and enhance algorithm improvement.

Table 1. Comparison of existing data generators.

Algorithms	high dimension	control density	control overlap	control arbitrary shape	randomization	high throughput	secondary editing
generateData [8]	No (only 2D)	Yes	No	No	No	Yes	No
Scikit-learn [9]	No (only 2D)	Yes	No	No	Partly	Yes	No
Elki [10]	Yes	Yes	No	No	Partly	No	Yes
OCLUS [4]	Yes	Partly	Yes	No	Partly	Yes	No
Qiu and Joe [11]	Yes	Partly	Yes	No	Partly	Yes	No
MDCGen [12]	Yes	Yes	Partly	No	Partly	Yes	No
Pei and Za¨ıane [13]	No (only 2D)	Partly	No	Partly	Partly	Yes	No
AC	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

2. Implementation

First, we present the following hypothesis:

1. The clusters in a dataset are always represented in a tree structure. More complex clusters are composed of smaller clusters. The clusters located at leaf nodes cannot be further divided, and we call them unit clusters (UCs).

2. The UC's samples follow an isotropic normal distribution $N(\mu, \sigma^2)$, which means that the samples in each dimension of the UC follow a normal distribution.

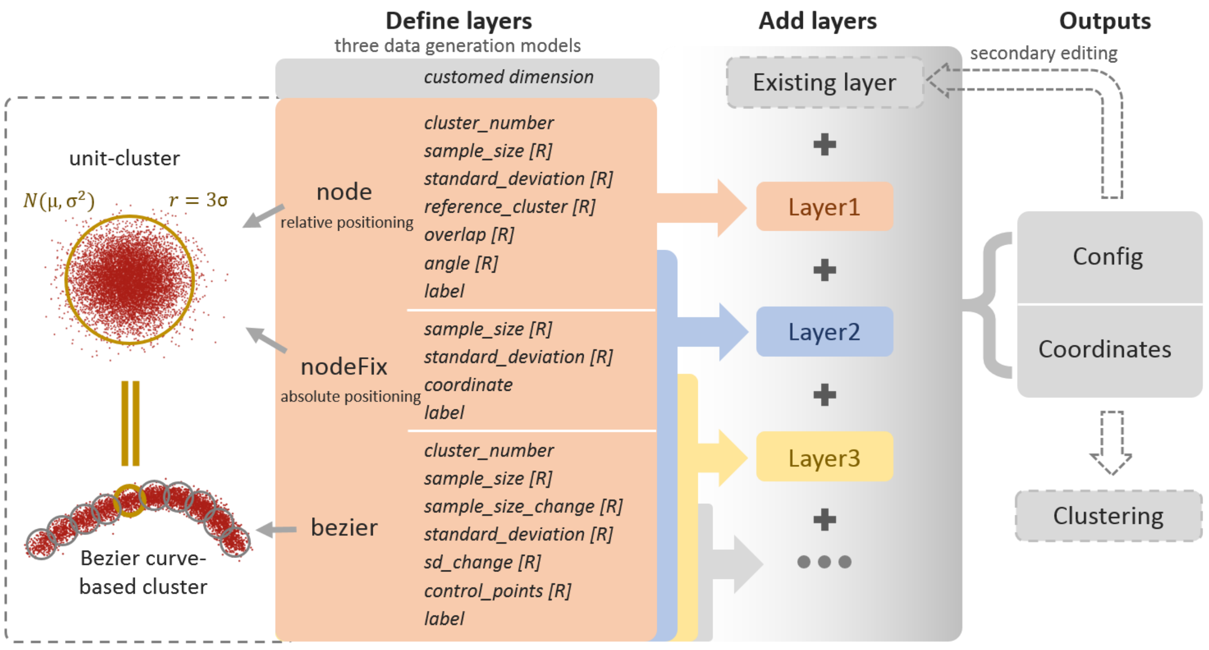

Fig. 1. AC logic diagram. [R] These parameters can be randomized. See "Document" at http://ac.fwgenetics.org

In fact, the first hypothesis is consistent with the idea of hierarchical clustering. In the second hypothesis, we specify that all the synthetic data follow a normal distribution. Some studies also introduced an additional distribution, such as a uniform or gamma distribution. Although the additional distribution enriched the data structure to some extent, we believe that a normal distribution is more consistent with real, ideal data. Therefore, the second hypothesis is reasonable when the included sample features are complete and independent and the sample size is large enough. All our synthesized clusters are based on UCs. Clusters with complex shapes are composed of multiple UCs. Accounting for the impact of the distribution characteristics on clustering algorithms, we define and control the following cluster characteristics: sample size, distribution size, cluster overlap, and shape. The density is determined by the sample size and distribution size. We used a layer-by-layer approach when building the dataset. One or more clusters can be added to each layer, and each layer has independent control over their clusters' characteristics. We provide three data generation models. Fig. 1 depicts the design logic.

2.1. Generation Model—node

node: Add UCs using relative positioning. First, a reference cluster (RC, the first cluster does not need a reference cluster) must be selected for the newly added UC (NC). Then, the NC is positioned by specifying the distance and angle of the NC relative to the RC. All added UCs, including the UCs inside Bezier curve-based clusters (BCs, see Section 2.3), can be selected as RCs. Considering that each BC consists of multiple (default 200) UCs, RCs should be numbered independently of the cluster labels. In a dataset, all UCs are numbered sequentially from 0-n. Each time one BC is added, the number of RCs increases by 200. By specifying the RC, a new UC can be added at any position using relative positioning (see Section 2.1.1 and 2.1.2). If no RC is specified, an existing UC will be randomly selected as the RC.

2.1.1. Overlap

The overlapping calculation is always done between two UCs. When an overlap of O is specified for a NC, it means that the overlap between the NC and the RC is equal to O, and the overlap between the NC and other existing UCs is not greater than O. According to the previous hypothesis, all UCs exhibit isotropic multivariate normal distributions with a radius of 3σ. Given a UC and an RC, which obey $N(\mu_1, {\sigma_1}^2)$ and $N(\mu_2, {\sigma_2}^2)$, respectively, when overlap is specified, the central distance between the UC and RC can be calculated using the Euclidean distance:

$$\begin{gathered} distance = max(\mathrm{a} + \mathrm{b} - \mathrm{overlap} \times \mathrm{a}, 0)\,, \\ a = min(3\sigma_1, 3\sigma_2)\,, b = max(3\sigma_1, 3\sigma_2) \end{gathered} \tag{1}$$

The value range of the overlap is not limited; the larger the value is, the closer the distance between UCs. When $overlap = 0$, the two UCs are exactly adjacent and the distance between the clusters $= A + B$. When the $overlap \geq (a + b)/a$, the two clusters overlap completely (that is, the central coordinates of the two UCs are the same).

2.1.2. Angle

After calculating the distance between the NC and RC according to the overlap, the central coordinates of the NC can be calculated by the angle. For multidimensional data, the input angle is a vector whose length is equal to the number of dimensions of the data. Each vector element corresponds to a counterclockwise rotation of the NC relative to the RC along each dimension (where the first dimension does not need to be rotated and is denoted as *0*). Using three-dimensional data as an example (Fig. 2), specifying that the angle vector $= (0, \alpha, \beta)$ results in rotating β counterclockwise along the y-axis and α counterclockwise along the z-axis. Given the centre coordinates of the RC (x_0, y_0, z_0) and the distances between UC and RC D, the centre coordinates of the NC can be calculated according to the angle:

$$\begin{pmatrix} z_i = z_0 + D\sin\alpha \\ y_i = y_0 + D\cos\alpha\sin\beta \\ x_i = x_0 + D\cos\alpha\cos\beta \end{pmatrix} \tag{2}$$

It should be noted that we previously defined the overlap value as the maximum overlap threshold between a new cluster and an existing non-reference cluster. When users specify the overlap and angle, existing non-reference clusters do not always meet the overlap threshold; this results in parameter conflicts. In this case, we added a parameter ($-cross$) to control whether clusters with conflicting parameters are shown (default is not displayed).

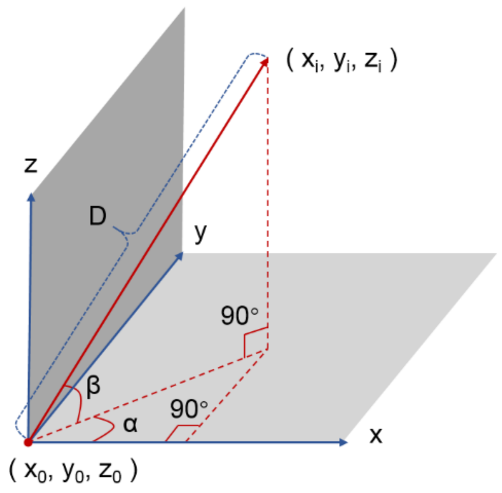

Fig. 2. Rotation angle of three-dimensional data.

2.2. Generation Model—nodeFix

nodeFix: Add UCs using absolute positioning. Since the UC is isotropic, we can directly determine its location by specifying the centre coordinates.

2.3. Generation Model—bezier

bezier: Add Bezier curve-based clusters (BCs). We support clusters distributed along Bezier curves of any order (including 2nd order, that is, straight lines). In theory, it is possible to design clusters with arbitrary shapes using Bezier curves. Users only need to specify the control points of the Bezier curve to generate a cluster distributed along a specific path. According to our hypothesis, a BC is composed of continuously distributed UCs, so the σ can also be used to control the BC's distribution size. To further expand the diversity of BCs, we provide the ability to control the dynamic change in the BC's distribution size and sample size along the Bezier curve. For example, the ending σ may be m (custom) times larger than the starting σ, the ending sample size may be n (custom) times larger than the starting sample size, or both may remain constant. In addition, we also allow the same cluster label to be assigned to multiple layers so that more UCs or BCs can be spliced to quickly design more complex clusters.

2.3.1. Bezier curve-based clusters

A BC is a cluster formed by arranging UCs along a specified Bezier curve of any order. When no Bezier curve is specified, a third-order Bezier curve is randomly generated. Each Bezier curve is specified as a list of control points and is represented as a vector consisting of the coordinates of each control point arranged in sequence. For example, in two-dimensional data, given three control points A(x_1,y_1), B(x_2,y_2), C(x_3,y_3), the Bezier curve = $(x_1,y_1,x_2,y_2,x_3,y_3)$. For a multi order Bezier curve consisting of n control points $(x_1,y_1,x_2,y_2,\cdots,x_n,y_n)$, we calculate the Bezier value for each dimension as:

$$\begin{pmatrix} Bezier_X = \sum_{i=0}^{n} \binom{n}{i} x_i (1-t)^{n-i} t^i \\ Bezier_Y = \sum_{i=0}^{n} \binom{n}{i} y_i (1-t)^{n-i} t^i \\ \cdots \end{pmatrix}, t \in [0,1] \tag{3}$$

After all the dimensions are calculated, the coordinates of any position on the Bezier curve can be obtained. Then, we select 200 points evenly along the Bezier curve's time axis and add a UC at each point. Selecting more points results in higher resolution and higher complexity. Here, the value 200 is a compromise between the resolution and the computational complexity.

2.3.2. Dynamic changes in BCs

The dynamic change in a BC is achieved by successively controlling each of the 200 UCs. The same strategy was used to dynamically control the sample size and σ. Using sample size control as an example, users can specify the ratio of the sample size of the 200th UC to the first UC, and the sample size of the remaining UCs is distributed according to arithmetic progression. Given a BC with a total sample size of S, the sample size of UCs from 1-200 (*S1-S200*) follow an arithmetic progression with tolerance d. Given the ratio $= r$, we define the sample size variations according to the following rules:

$$\begin{cases} S_1 = S_{200}, & r = 0 \\ S_{200} = S_1 + rS_1, & r > 0 \\ S_1 = S_{200} + |r|S_{200}, & r < 0 \end{cases} \tag{4}$$

After all the dimensions are calculated, the coordinates of any position on the Bezier curve can be obtained. Then, we select 200 points evenly along the Bezier curve's time axis and add a UC at each point. Selecting more points results in higher resolution and higher complexity. Here, the value 200 is a compromise between the resolution and the computational complexity.

The number of UCs inside a BC is $n = 200$. The tolerance and the sample size of the first UC can be deduced from the arithmetic progression formula.

$$\begin{cases} S_1 = \dfrac{S}{N}, & r = 0 \\ S_1 = \dfrac{2S}{n(2+r)}, d = \dfrac{2Sr}{n(n-1)(2+r)}, & r > 0 \\ S_1 = \dfrac{2S(1-r)}{n(2-r)}, d = \dfrac{2Sr}{n(n-1)(2-r)}, & r < 0 \end{cases} \tag{5}$$

In this way, we can obtain the theoretical sample size of each UC. Similarly, we can also calculate the dynamic σ for each UC. In practice, arithmetic progression often contains many non-integers, but the sample size must be a positive integer. To guarantee the accuracy of the total sample size, we use an independent variable m to record the accumulated value of the fractional components of the sample sizes from all previous UCs. For the sample size of the current UC $a.b$ (where a is the integer component and b is the fractional component), we first add m to the current fractional component:

$$m = m + b \tag{6}$$

Then, proceed as follows according to m

$$\begin{cases} a = a + 1, m = m + 1, & m \geq 1 \\ a = a, m = m, & m < 1 \end{cases} \tag{7}$$

Use a as the final sample size for the current UC. It should be noted that for the last unit cluster, we cannot simply use *m* to determine whether 1 should be added to the integer component. We must add a to the sample size of all the added UCs, compare a with the specified total sample size, and then add 1 only if the total sample size is not reached.

2.4. Randomization

To evaluate the robustness of a clustering algorithm, we introduce the parameter randomization function. In the simplest case, users only need to specify the number of UCs and BCs to randomly generate clusters with various complex structures. The flexible randomization design allows users to control some cluster characteristics while randomizing others, which can be used to evaluate the robustness of clustering algorithms on various datasets with different characteristic combinations.

To keep the global distribution within a reasonable range, we randomize different parameters over different ranges.

2.4.1. Add UCs

Select the RC, where *clusterNum* is the number of added UCs:

$$RC = round(rand(0,1)clusterNum) \tag{8}$$

Standard deviation:

$$\sigma = 1 + 10rand(0,1) \tag{9}$$

Sample size:

$$sampleSize = 300\big(1 + \sigma \times rand(0,1)\big) \tag{10}$$

Angle vector:

$$angle_i = 360rand(0,1) \tag{11}$$

Overlap:

$$overlap = \begin{cases} 0.7rand(0,1), & rand(0,1) \geq 0.5 \\ -rand(0,1), & otherwise \end{cases} \tag{12}$$

2.4.2. Add BCs

Initial standard deviation:

$$\sigma = 2.2 - 2.1rand(0,1) \tag{13}$$

Initial sample size:

$$sampleSize = 300\big(1 + \sigma \times rand(0,1)\big) \tag{14}$$

Coordinate vector of control points:

$$control_i = 30\,max(1, \sigma)\,rand(0,1) \tag{15}$$

Ratio of sample size:

$$rss = \begin{cases} 0, & \text{if } rand(0,1) < 0.3 \\ 10rand(0,1), & \text{else if } rand(0,1) < 0.5 \\ -10rand(0,1), & \text{else} \end{cases} \tag{16}$$

Ratio of σ:

$$rsd = \begin{cases} 0, & \text{if } rand(0,1) < 0.3 \\ 5rand(0,1), & \text{else if } rand(0,1) < 0.5 \\ -5rand(0,1), & \text{else} \end{cases} \tag{17}$$

When randomly adding a new BC, the maximum overlap between each UC inside the BC and other existing UCs is 0. If the overlap threshold cannot be satisfied, the new BC will be translated iteratively. The translation parameter -offset is a vector whose length is equal to the number of dimensions, and the new BC will translate in each dimension according to the corresponding element. To reasonably control translational amplitude and computational complexity, translation randomization should consider translational direction, the number of iterations, the number of dimensions and the initial σ. Therefore, we specify each translation as (calculated separately for each dimension):

$$offset_i = \begin{cases} \dfrac{DN \times \sigma \times IN}{8}, & \text{if } rand(0,1) > 0.5 \\ -\dfrac{DN \times \sigma \times IN}{8}, & \text{else} \end{cases}, \qquad (18)$$

DN: dimension number, IN: iteration number

To reproduce and edit the data more conveniently, a configuration file will be output at the same time during each run. AC can directly accept the configuration file and allow new clusters to be added.

3. Examples

As an example, Fig. 3 shows AC's ability to control cluster characteristics. Fig. 3a shows a decrease in density caused by a constant σ and decreasing sample size. Fig. 3b shows a decrease in density caused by a decreasing σ and constant sample size. Fig. 3c shows an increase in overlap. In fact, the σ, sample size and overlap parameters do not affect each other, so more complex clusters can be designed by combining parameters. Figs. 3d-g are comprehensive displays. Fig. 3d shows the comprehensive control of UCs and BCs. On the left is a spiral composed of 16 different UCs, arranged sequentially at a gradual angle. The right side shows seven morphological transformations of the same BC with the total sample size unchanged. From left to right are: 1. The initial BC. 2. The BC with an increased σ. 3. The BC as the σ increases further. 4. The BC with a decreasing sample size and increasing σ. 5. The BC with an increasing sample size and increasing σ. 6. The BC with a decreasing sample size and decreasing σ. 7. The BC with an increasing sample size and decreasing

σ. Finally, a very low-density BC (second-order Bezier curve) distributed along the horizontal axis was added as noise. Fig. 3e simulates a handwritten "Bezier" consisting of 10 BCs and 1 UC. Although a total of 11 clusters have been added, only 6 colours are shown, because some BCs have been assigned to the same label by the -label parameter. For example, the letter "B" is actually made up of three BCs. This splicing reduces the difficulty of designing complex Bezier curves. In addition, by dynamically controlling the BCs' σ and sample size, even the details of pen pressure changes during writing can be reflected. Fig. 3f shows randomization of the default parameters and consists of 2 BCs and 5 UCs; these clusters were regenerated 5 times. The random characteristics of the 5 groups of clusters are significantly different. Fig. 3g simulates a stick figure of a lantern composed of 20 BCs and 5 UCs and was assigned 12 labels by splicing the clusters. The parameter configuration files for all the above examples are available on our website. These examples can be reproduced by importing the configuration files into AC. More examples can be found at http://ac.fwgenetics.org

It is worth noting that some data generators put forward the addition of noise or outliers. However, this is not specifically designed for AC because any type of outlier can be easily implemented by adding extremely low-density clusters. In short, through the combination of UCs and BCs, AC can be used to not only design complex clusters but also to accurately control design details. The parameter randomization provides the possibility for batch generation of benchmark datasets. These functions sufficiently fulfil the various evaluation requirements of clustering algorithms.

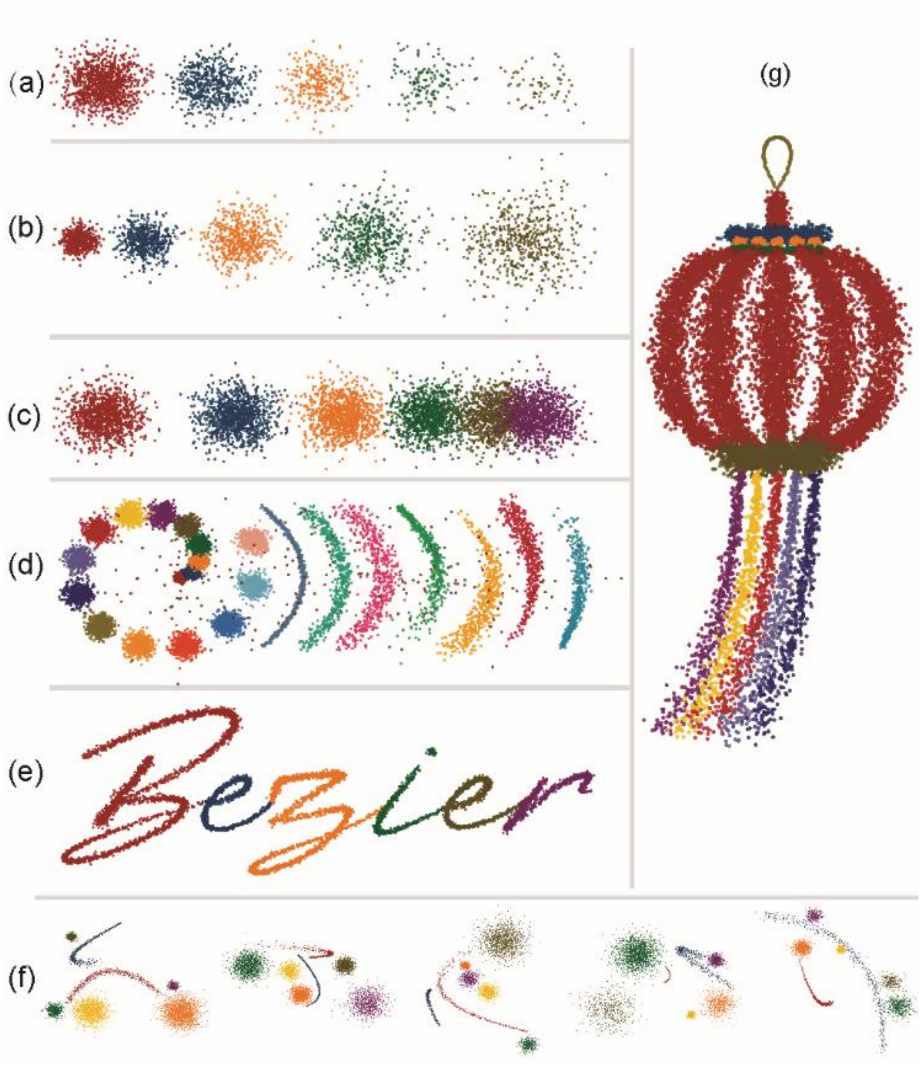

Fig. 3. Example of controlling cluster characteristics. a-g, See Section 3 for details.

4. Experiments

To verify the effectiveness of AC on clustering performance evaluation, we designed a parameter set containing 4 groups of characteristic combinations to generate benchmark datasets. See Table 2 for details. The parameter set configuration files are available on our website. We recommend that users design their own parameter sets for specific purposes. Each group of characteristic combinations generates 10 datasets used to evaluate clustering algorithm robustness. Then, we evaluated 9 popular clustering algorithms, including K-Means [14], DBSCAN [15], OPTICS [16], Hierarchical (Ward algorithm) [17], MoG [18], Spectral [19], Subspace [20], AP [21] and DensityPeak [22]. All algorithms were implemented using FCPS [23]. Several clustering performance evaluation metrics were calculated using scikit-learn [9], including the external index: Rand index [24], Mutual Information based scores [25], Homogeneity, completeness, V-measure [26] and Fowlkes-Mallows scores [27]; and the internal index: Silhouette Coefficient [28], Calinski-Harabasz Index [29] and Davies-Bouldin Index [30]. See supplementary material for datasets, config files and clustering results.

Table 2. Example of parameter set used for benchmarking.

Group	Assess target	Design
Density	different density	5UCs[*]+5UCs+5Ucs: constant overlap, increasing sample size, random σ[#], random RC[@], random angle
Overlap	different overlap	5UCs+5UCs+5UCs: constant sample size, constant σ, increasing overlap, random RC, random angle
Shape	non-convex shape	10BCs[$]: constant sample size, constant σ, random control points
Complexity	comprehensive evaluation	2BCs+5UCs+3BCs+5UCs: all random

[*]UC – Unit Cluster; [#]σ – standard deviation; [@]RC – Reference Cluster; [$]BC – Bezier curve-based Cluster

4.1. Parameter optimization for clustering

Some clustering algorithms are sensitive to parameters and need to be optimized to obtain reasonable results. Of the nine algorithms we used, K-Means, Hierarchical (Ward algorithm), MoG, Spectral, and Subspace only need to specify the number of clusters; no other optimization is required. DBSCAN, OPTICS, AP and DensityPeak do not need to specify the number of clusters, but they need to optimize other parameters. The density-based clustering algorithms DBSCAN and OPTICS have two sensitive parameters: *Radius* and *minPts*.

In practice, we cannot adjust the parameters repeatedly according to the results. Therefore, we set these two parameters to $NULL$ to invoke the parameter estimation feature in FCPS. In AP, the number of clusters primarily depends directly on the *ExemplarPreferences* parameter. To ensure the number of clusters is closer to the actual result, we set *ExemplarPreferences* to 0. The key to DensityPeak is to select appropriate thresholds for rho and $delta$. Manual selection is not suitable for batch clustering due to inefficiency and difficulty for complex data. Although the product method is popular, our tests revealed that the product method has poor performance on the density and overlap groups, whether normalized or not. After continued testing, we discovered a better strategy: 1. For the two-dimensional array of rho and $delta$, sort rho from large to small and save the sequence number (1-index) in a new column $rhoNum$; 2. Sort $delta$ from large to small and save the sequence number (1-index) in a new column $deltaNum$; 2. Sort the product of $rhoNum$ and $deltaNum$ from large to small. 3. Given the number of target clusters n, use the maximum value of the top n rho as the rho threshold and use the maximum value of the top n $delta$ as the $delta$ threshold. Although this strategy sometimes results in slightly more than n clusters, the overall performance is satisfactory and stable on a variety of complex data.

4.2. Evaluation

Fig. 4 shows a visualization of clustering from one dataset (Fig. 4a) and the evaluation on all 10 datasets (Fig. 4b). As seen from the results, no single algorithm achieved satisfactory results on all four groups of characteristic combinations. In general, the performance on convex data (the density and overlap groups) was better than that on nonconvex data (the shape and complexity groups). The clustering error for convex data typically originated from an algorithm's inability to detect or distinguish certain clusters (error 1), while the clustering error for nonconvex data occurred to division and reorganization within clusters (error 2). Error 2 had a greater impact on the results because it destroys the integrity of a single cluster. For the complexity group, almost no algorithm could avoid error 2. Unexpectedly, Subspace was prone to error 2 for both convex and nonconvex data. Relatively speaking, Spectral had a lower error rate and generally outperformed other algorithms. However, the computational complexity of Spectral was much higher (Table 3), which limits its application. AP and DensityPeak were balanced between performance and complexity. K-Means and MoG performed well, but the risk of K-Means introducing error 2 was higher, and MoG was less stable. The density-based algorithms DBSCAN and OPTICS were sensitive to parameters. They obtained poor results with estimated parameters. Although better results could be obtained by repeatedly adjusting the parameters according to the ground truth classes, this level of control is impossible to achieve in practical applications. Hierarchical (Ward) performed poorly on the density, overlap and complexity groups, but surprisingly, it outperformed other algorithms on the shape group, which may be informative when developing new clustering algorithms. In addition, we noticed that even though some algorithms perform well on some evaluation metrics, the actual clustering was not ideal. Additionally, there was still a large gap between the real value and the best expected value for the three internal indices on the ground truth classes. All of these results suggest that the metrics for clustering performance evaluation need to be further optimized.

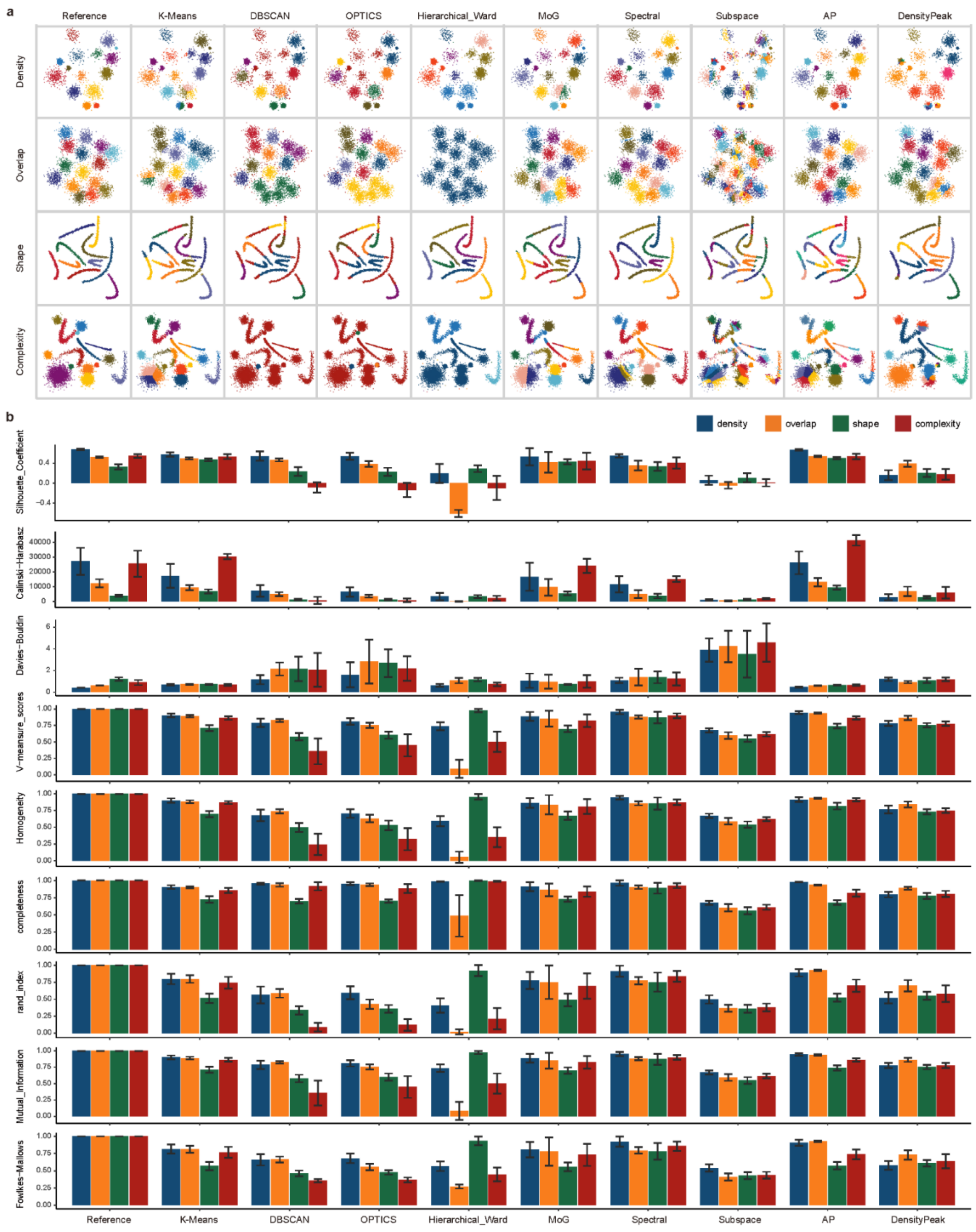

Fig. 4. Performance of the nine clustering algorithms on datasets comprising four characteristic combinations. (a), Visualization of clustering from one dataset. (b), The performance evaluation metrics for all 10 datasets. Error bars represent the standard deviation.

Table 3. The average time consumption (seconds) of each algorithm on each dataset.

Algorithms	Shape	Density	Overlap	Complexity
K-Means	0.3±0	0.3±0	0.3±0	0.6±0.1
DBSCAN	10.4±0.8	9.8±0.5	3.5±0.3	26.2±1.6
OPTICS	4±0.3	3.3±0.2	3.3±0.3	14.2±0.3
Hierarchical	2.2±0.2	1.9±0.1	2.1±0.2	12.8±4.5
MoG	0.4±0.1	0.4±0.1	0.4±0.1	0.8±0.1
Spectral	**9294.8±1271.7**	**6641.5±777.9**	**6672±723.8**	**118080.2±51921.6**
Subspace	5.1±0.3	6.8±0.2	6.7±0.3	15.9±2.2
AP	149.7±46.1	156.2±37.2	96.5±17.6	1490.9±707.1
DensityPeak	1116.5±438.4	941.1±225.6	927.2±177	12448.6±2858

5. Conclusion

In this paper, we introduced AC, a powerful data generator for clustering. AC makes it easy to generate data with complex structures by introducing the concept of unit clusters. The ability to customize arbitrary density, arbitrary overlap, and arbitrary shape clusters was achieved for the first time in AC, filling the gap of benchmark data in the evaluation of clustering algorithms. AC provides the following advantages:

1. AC generates high-quality data without interference from data acquisition and preprocessing with regard to quality control, standardization, and feature selection, for example. All data generated by AC are points distributed in n-dimensional space, and the Euclidean distance can fully represent the distance between these

points. Therefore, these datasets can be used to more purely evaluate the performance of the clustering algorithm itself.

2. AC can thoroughly customize the cluster characteristics. The characteristics that affect the performance of clustering algorithms include different densities, overlaps, and complex shapes (mainly nonconvex). AC can control and combine these characteristics at will and is much more flexible than other data generators. This will help clustering algorithms perform a variety of targeted tests.

3. By randomizing, AC can generate a large number of datasets that conform to specified characteristics. The main parameters of AC are randomly generated within a certain range by default. For example, the position of the cluster is random when the angle is not specified, the density of the cluster is random when the sample size and σ are not specified, and the shape of the Bezier curve-based cluster is random when the control points are not specified. Therefore, AC provides sufficient data to evaluate the robustness of clustering algorithms.

4. The config file of AC can be used to conveniently edit the existing data. AC supports three ways to save data: raw parameters (retain randomization, rerun will generate new data), fixed parameters (no randomization, rerun will generate exactly the same data as before), and data file (containing the coordinates and labels of each point). Both raw and fixed parameters are kept in a config file and are easy to modify and use. This improves the efficiency of data design and facilitates data dissemination.

As an example, we designed a set of benchmark data and tested the performance of nine popular clustering algorithms. The results clearly show the advantages and disadvantages of these algorithms and suggest that there are still many deficiencies in the current clustering algorithms. In future research on clustering algorithms, AC is expected to generate enough targeted data to assist in the design of new algorithms.

AC is based on the sensitivity of existing clustering algorithms to data, and all parameters are limited to the existing knowledge of clustering performance evaluation. If new clustering algorithms need new data characteristics to be supported in the future, then we will further enrich the function of AC. In addition, the randomization range of parameters is built-in, and we will decide whether to provide control over randomization based on user feedback.

Acknowledgment

This work was supported by the Fundamental Research Funds for the Central Universities (3332020019).

References

[1] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O.P. Patel, A. Tiwari, M.J. Er, W. Ding, C.-T. Lin, A review of clustering techniques and developments, Neurocomput., 267 (2017) 664–681.

[2] F. Fang, L. Qiu, S. Yuan, Adaptive core fusion-based density peak clustering for complex data with arbitrary shapes and densities, Pattern Recognition, 107 (2020) 107452.

[3] Y. Zhu, K.M. Ting, M.J. Carman, M. Angelova, CDF Transform-and-Shift: An effective way to deal with datasets of inhomogeneous cluster densities, Pattern Recognition, 117 (2021) 107977.

[4] D. Steinley, R. Henson, OCLUS: An Analytic Method for Generating Clusters with Known Overlap, J Classif, 22 (2005) 221-250.

[5] B.K. Patra, S. Nandi, P. Viswanath, A distance based clustering method for arbitrary shaped clusters in large datasets, Pattern Recognition, 44 (2011) 2862-2870.

[6] P. Fränti, S. Sieranoja, K-means properties on six clustering benchmark datasets, Applied Intelligence, 48 (2018) 4743-4759.

[7] Z. Ming, C. Luo, W. Gao, R. Han, Q. Yang, L. Wang, J. Zhan, BDGS: A Scalable Big Data Generator Suite in Big Data Benchmarking, Springer International Publishing, Cham, 2014, pp. 138-154.

[8] N. Fachada, A.C. Rosa, generateData—A 2D data generator, Software Impacts, 4 (2020) 100017.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, Scikit-learn: Machine Learning in Python, J Mach Learn Res, 12 (2011) 2825-2830.

[10]E. Schubert, A. Koos, T. Emrich, A. Züfle, K.A. Schmid, A. Zimek, A framework for clustering uncertain data, Proc. VLDB Endow., 8 (2015) 1976–1979.

[11]W. Qiu, H. Joe, Generation of Random Clusters with Specified Degree of Separation, J Classif, 23 (2006) 315-334.

[12]T. Zseby, D. Ferreira, A. Zimek, MDCGen: Multidimensional Dataset Generator for Clustering, J Classif, 36 (2019) 599-618.

[13]Y. Pei and O. Zaïane, A synthetic data generator for clustering and outlier analysis, University of Alberta Edmonton, AB, Canada. (2006).

[14]J.A. Hartigan, M.A. Wong, Algorithm AS 136: A K-Means Clustering Algorithm, Journal of the Royal Statistical Society. Series C (Applied Statistics), 28 (1979) 100-108.

[15]M. Ester, H.P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, in: International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[16]M. Ankerst, M.M. Breunig, H.P. Kriegel, J. Sander, OPTICS: Ordering Points To Identify the Clustering Structure, Proceedings of the 1999 ACM SIGMOD international conference on Management of data, (1999) 49-60.

[17]J.J. Ward, Hierarchical grouping to optimize an objective function, Journal of the American Statistical Association, 58 (1963) 236-244.

[18]W.C. Chen, R. Maitra, V. Melnykov, EMCluster: EM Algorithm for Model-Based Clustering of Finite Mixture Gaussian Distribution, R Package, http://cran.r-project.org/package=EMCluster, (2015).

[19]A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, Advances in Neural Information Processing Systems 14, (2001) 849--856.

[20]C.C. Aggarwal, P.S. Yu, Finding Generalized Projected Clusters in High Dimensional Spaces, Acm Sigmod Record, 29 (2000) 70-81.

[21]B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science, 315 (2007) 972-976.

[22]A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science, 344 (2014) 1492-1496.

[23]M.C. Thrun, Q. Stier, Fundamental clustering algorithms suite, SoftwareX, 13 (2021) 100642.

[24]D. Steinley, Properties of the Hubert-Arabie Adjusted Rand Index, Psychol Methods, 9 (2004) 386-396.

[25]N.X. Vinh, J. Epps, J. Bailey, Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance, J Mach Learn Res, 11 (2010) 2837-2854.

[26]A. Rosenberg, J. Hirschberg, V-Measure: A conditional entropy-based external cluster evaluation, Conference on Emnlp-conll2007, pp. 410-420.

[27]D.W. Turner, A Method for Comparing Two Hierarchical Clusterings, Journal of the American Statistical Association, 78 (1983) 583.

[28]P.J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics, 20 (1987) 53-65.

[29]T. Calinski, J. Harabasz, A dendrite method for cluster analysis, Commun Stat Theory Methods, 3 (1974) 1-27.

[30]D.L. Davies, D.W. Bouldin, A Cluster Separation Measure, IEEE transactions on pattern analysis and machine intelligence, PAMI-1 (1979) 224-227.