Software Impacts 4 (2020) 100017

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

generateData—A 2D data generator

Nuno Fachada [a,b,*], Agostinho C. Rosa [b]

[a] HEI-Lab — Digital Human-Environment Interaction Lab, Lusófona University, Lisbon, Portugal
[b] Institute for Systems and Robotics (ISR/IST), LARSyS, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal

ARTICLE INFO

Keywords:
Data generation
2D
Clustering

ABSTRACT

`generateData` is a MATLAB/Octave function for generating 2D data clusters. Data is created along straight lines, which can be more or less parallel depending on the selected input parameters. The function also allows to fine-tune the generated data with respect to number of clusters, total data points, average cluster separation and several other distributional properties.

Code metadata

Current code version	v2.0.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-D-20-00014
Legal Code License	MIT License
Code versioning system used	Git
Software code languages, tools, and services used	MATLAB/GNU Octave
Compilation requirements, operating environments & dependencies	MATLAB ≥ R2011a or GNU Octave ≥ 4.0.0
If available Link to developer documentation/manual	https://github.com/fakenmc/generateData/blob/v2.0.0/README.md
Support email for questions	nuno.fachada@ulusofona.pt

1. Introduction

Massive amounts of data are being produced everyday, raising a number of challenges on its storing and processing [1–3]. However, there are still many instances where, for the intended purposes, data is insufficient and/or expensive — thus making synthetic data generation an appealing alternative [4–7]. One such instance is data generation for evaluating clustering algorithms [8,9].

In this paper we discuss the impact of `generateData`, a MATLAB [10] and GNU Octave [11] function for generating 2D data primarily aimed at testing clustering algorithms. Section 2 offers an overview of how the data generation algorithm works as well as some output examples. The impact of `generateData` is presented in Section 3. The limitations of this work, as well as potential improvements are discussed in Section 4.

2. Description

`generateData` is a MATLAB/Octave function for generating 2D data clusters. The function allows to fine-tune several characteristics of the generated data through a number of required and optional parameters, summarized in Tables 1 and 2, respectively. In any case, data is created along straight lines. The exact angle of these lines with respect to the x axis is drawn from the normal distribution. The mean and standard deviation of this distribution correspond to parameters `angleMean` and `angleStd` in Table 1. The latter influences how parallel are the lines supporting the data. A standard deviation of zero yields completely parallel lines, while higher values increasingly randomize line orientation. In turn, line length is drawn from the folded normal distribution, with mean and standard deviation given as parameters `lengthMean` and `lengthStd`, respectively.

The method for placing points around lines is as follows. First, a projection of each point on the line is obtained from the distribution specified in optional parameter `pointDist`. The default is the uniform distribution, i.e., point projections are placed evenly along the line. Using the normal distribution, the line center is used as the mean, with the line length corresponding to 3 standard deviations — thus, there is a small chance projections are placed outside the line. After the point projections are determined, points are either placed around their projections using the bivariate normal distribution (a 2D placement, the

* Corresponding author at: HEI-Lab — Digital Human-Environment Interaction Lab, Lusófona University, Lisbon, Portugal.
E-mail addresses: nuno.fachada@ulusofona.pt (N. Fachada), acrosa@laseeb.org (A.C. Rosa).

https://doi.org/10.1016/j.simpa.2020.100017
Received 8 May 2020; Received in revised form 13 May 2020; Accepted 21 May 2020

2665-9638/© 2020 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Table 1
Required parameters.

Parameter	Description
`angleMean`	Mean angle in radians of the lines on which clusters are based. Angles are drawn from the normal distribution.
`angleStd`	Standard deviation of line angles.
`numClusts`	Number of clusters (and therefore of lines) to generate.
`xClustAvgSep`	Average separation of line centers along the X axis.
`yClustAvgSep`	Average separation of line centers along the Y axis.
`lengthMean`	Mean length of the lines on which clusters are based. Line lengths are drawn from the folded normal distribution.
`lengthStd`	Standard deviation of line lengths.
`lateralStd`	Cluster "fatness", i.e., the standard deviation of the distance from each point to its projection on the line. The way this distance is obtained is controlled by the optional `pointOffset` parameter.
`totalPoints`	Total points in generated data. These will be randomly divided between clusters using the half-normal distribution with unit standard deviation.

default), or on a second line, perpendicular to the original one, using a normal distribution (a 1D placement). In either case, the projection is used as the mean value, while the standard deviation is given by the `lateralStd` parameter. The type of placement, 1D or 2D, is defined by the optional `pointOffset` parameter.

Fig. 1 shows four datasets created with `generateData` using the parameters given in Table 3.

3. Impact

The `generateData` script was originally created to test the AMVIDC clustering algorithm [12]. The algorithm performs agglomerative hierarchical clustering using minimum volume increase and minimum direction change clustering criteria, and was inspired by the typical layout of spectrometric data after being processed with principal component analysis (PCA). More specifically, the PCA score plots of spectrometric data were found to exhibit distinct groups scattered along a preferential direction, forming low volume clusters. The `generateData` script was designed to generate data of this kind, offsetting the lack of actual experimental data and thus allowing to better tune and test the AMVIDC algorithm.

In reference [13], Zamberletti et al. "investigated the role of individual wetlands within a wetlandscape in sustaining an amphibian population". Wetlandscapes – sets of multiple hydrologically connected wetlands – were modeled as networks, with nodes representing individual wetlands, and connections corresponding to flows of organisms. Zamberletti et al. used `generateData` to create a random network of wetlands for running a population dynamics model. The script was suited to this problem, since it allowed to define deterministic topological network parameters (e.g., number of clusters), while adding some random variation via its stochastic arguments (e.g., average cluster separation).

Table 3
Parameters used for the examples shown in Fig. 1.

Parameter	Fig. 1(a)	Fig. 1(b)	Fig. 1(c)	Fig. 1(d)
Seed for `rng()`	1111	1111	123	123
`angleMean`	$\pi/4$	$-\pi/2$	$\pi/2$	$\pi/2$
`angleStd`	$\pi/16$	$\pi/10$	0	0
`numClusts`	5	6	4	4
`xClustAvgSep`	10	15	2	2
`yClustAvgSep`	10	15	2	2
`lengthMean`	12	15	4	4
`lengthStd`	4	10	1	1
`lateralStd`	1	2	0.1	0.1
`totalPoints`	800	2500	2500	2500
`pointDist`	`'norm'`	`'unif'`	`'unif'`	`'norm'`
`pointOffset`	`'1D'`	`'2D'`	`'1D'`	`'2D'`

With the goal of evaluating D3CAS, a dynamical and big data-oriented clustering algorithm for processing data streams, Molina & Hasperué [14,15] used `generateData` to create datasets with up to 100 000 points, an adequate size for their testing requirements.

Alabdulatif et al. [16–18] made use of `generateData` for a series of investigations on cloud and edge computing privacy-related data analytics. Datasets were generated with the purpose of evaluating distributed and privacy-preserving versions of several clustering algorithms in a number of different scenarios.

In reference [19], Hao et al. presented a video summarization approach consisting of generating a short video summary while maintaining the overall meaning of the original video. The approach worked by applying sparse subspace clustering with automatically estimated number of clusters to deep features of objects in key-frames. `generateData` was used to produce synthetic data for testing the accuracy of the method for estimating the number of clusters.

Olukanmi et al. [20,21] used `generateData` to assemble scenarios with one million data points with the purpose of assessing the proposed k-means-lite and k-means-lite++ clustering algorithms — highly scalable versions of their non-lite counterparts.

4. Limitations and potential improvements

The main drawback of `generateData` is obviously being limited to 2D. Nonetheless, the basic ideas of how data is generated are extendable to n-D, with several parameters such as angle mean/standard deviation and average separation by axis being given as vectors instead of scalars. Another potential limitation is that the function is only available for either the MATLAB or GNU Octave environments. The former is a proprietary offering, potentially inaccessible to many researchers, while the latter is a worthwhile open source, largely compatible alternative. However, languages such as Python, R and Julia have been gaining popularity in the scientific computing community [22,23]. As such, one of our goals is to port `generateData` to these languages, making it available to a wider audience.

Table 2
Optional named parameters.

Name	Default	Description
`allowEmpty`	`false`	Allow empty clusters?
`pointDist`	`'unif'`	Specifies the distribution of points along lines, with two possible values: (1) `'unif'` distributes points uniformly along lines; or, (2) `'norm'` distributes points along lines using a normal distribution (line center is the mean and line length is equal to 3 standard deviations).
`pointOffset`	`'2D'`	Controls how points are created from their projections on the lines, with two possible values: (1) `'1D'` places points on a second line perpendicular to the cluster line using a normal distribution centered at their intersection; or, (2) `'2D'` places point using a bivariate normal distribution centered at the point projection.

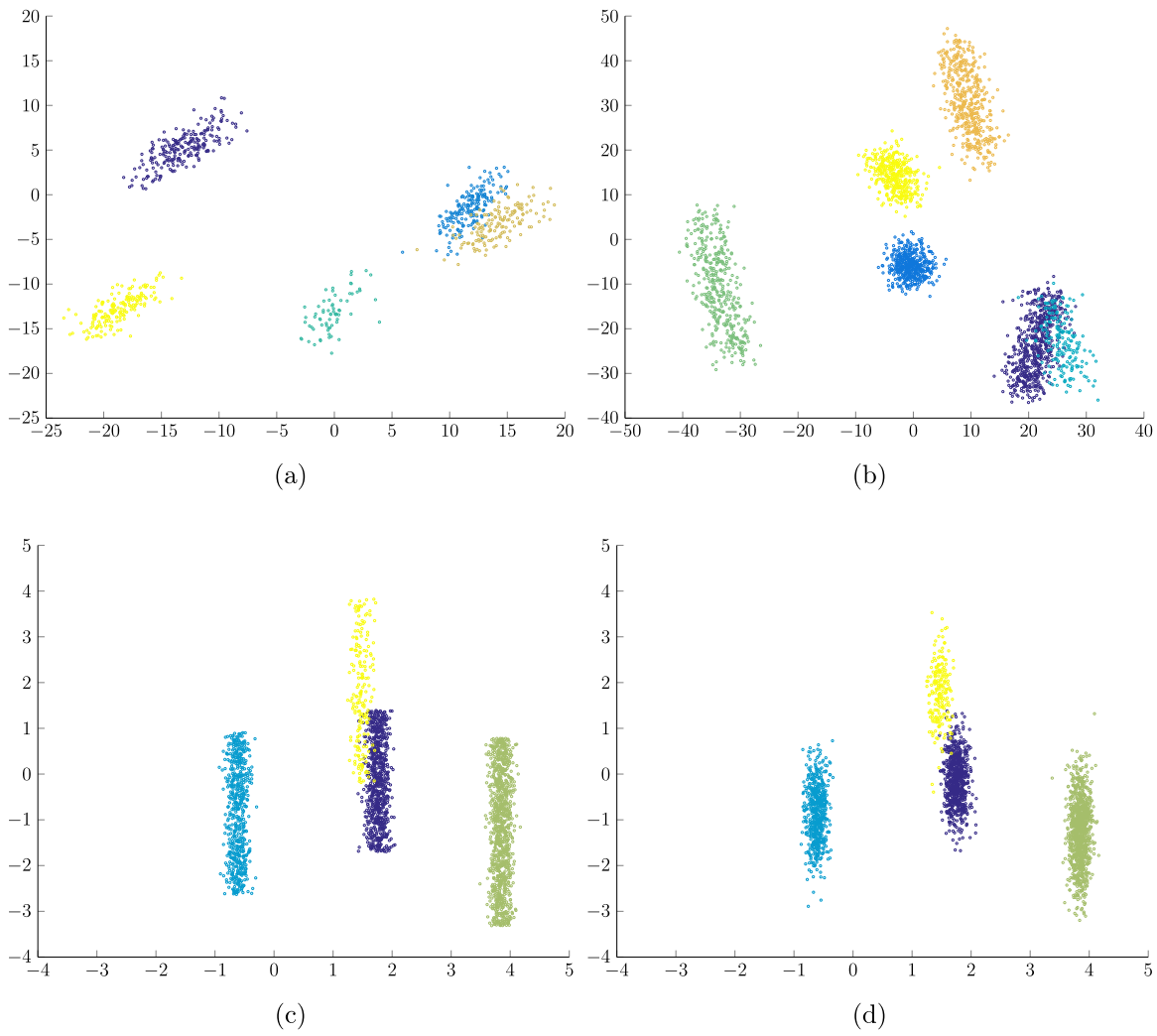

Fig. 1. Examples of datasets created with `generateData` in MATLAB using various parameters. Parameters for each figure are shown in Table 3. The `allowEmpty` parameter was always set to `false`.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Funding: This work was supported by Fundação para a Ciência e Tecnologia (FCT), Portugal grant UIDB/05380/2020 (HEI-Lab); and the LARSyS – FCT Plurianual, Portugal funding 2020–2023.

References

[1] A. L'Heureux, K. Grolinger, H.F. Elyamany, M.A.M. Capretz, Machine learning with big data: Challenges and approaches, IEEE Access 5 (2017) 7776–7797, http://dx.doi.org/10.1109/ACCESS.2017.2696365.

[2] M. Younas, Research challenges of big data, SOCA 13 (2) (2019) 105–107, http://dx.doi.org/10.1007/s11761-019-00265-x.

[3] W. Ejaz, A. Anpalagan, Dimension reduction for big data analytics in Internet of Things, in: Internet of Things for Smart Cities: Technologies, Big Data and Security, Springer, 2019, pp. 31–37, http://dx.doi.org/10.1007/978-3-319-95037-2_3.

[4] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, S. Birchfield, Training deep networks with synthetic data: Bridging the reality gap by domain randomization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 969–977, http://dx.doi.org/10.1109/CVPRW.2018.00143.

[5] A. Kar, A. Prakash, M.-Y. Liu, E. Cameracci, J. Yuan, M. Rusiniak, D. Acuna, A. Torralba, S. Fidler, Meta-Sim: Learning to generate synthetic datasets, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 4551–4560, http://dx.doi.org/10.1109/ICCV.2019.00465.

[6] A. Anaby-Tavor, B. Carmeli, E. Goldbraich, A. Kantor, G. Kour, S. Shlomov, N. Tepper, N. Zwerdling, Not enough data? Deep learning to the rescue! 2019, arXiv preprint arXiv:1911.03118. https://arxiv.org/abs/1911.03118.

[7] J. Li, L. Qiu, B. Tang, D. Chen, D. Zhao, R. Yan, Insufficient data can also rock! Learning to converse using smaller data with augmentation, in: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, in: AAAI-19, vol. 33, 2019, pp. 6698–6705, http://dx.doi.org/10.1609/aaai.v33i01.33016698.

[8] Y. Pei, O. Zaïane, A Synthetic Data Generator for Clustering and Outlier Analysis, Tech. Rep., (TR06-15) University of Alberta, 2006, http://dx.doi.org/10.7939/R3B23S.

[9] M. Zaït, H. Messatfa, A comparative study of clustering methods, Future Gener. Comput. Syst. 13 (2) (1997) 149–159, http://dx.doi.org/10.1016/S0167-739X(97)00018-6.

[10] MATLAB R2020a, MathWorks, Natick, Massachusetts, USA, 2020, https://www.mathworks.com/products/matlab.html.

[11] J.W. Eaton, D. Bateman, S. Hauberg, R. Wehbring, GNU Octave version 5.2.0 manual: A high-level interactive language for numerical computations, 2020, https://www.gnu.org/software/octave/doc/v5.2.0/.

[12] N. Fachada, M.A.T. Figueiredo, V.V. Lopes, R.C. Martins, A.C. Rosa, Spectrometric differentiation of yeast strains using minimum volume increase and minimum direction change clustering criteria, Pattern Recognit. Lett. 45 (2014) 55–61, http://dx.doi.org/10.1016/j.patrec.2014.03.008.

[13] P. Zamberletti, M. Zaffaroni, F. Accatino, I.F. Creed, C. De Michele, Connectivity among wetlands matters for vulnerable amphibian populations in wetlandscapes, Ecol. Model. 384 (2018) 119–127, http://dx.doi.org/10.1016/j.ecolmodel.2018.05.008, http://www.sciencedirect.com/science/article/pii/S0304380018301686.

[14] R. Molina, Estudio e implementación de una técnica de clustering dinámico para trabajar con flujos de datos, Universidad Nacional de La Plata, La Plata, Argentina, 2018, http://sedici.unlp.edu.ar/handle/10915/82400.

[15] R. Molina, W. Hasperué, D3CAS: un Algoritmo de Clustering para el Procesamiento de Flujos de Datos en Spark, in: XXIV Congreso Argentino de Ciencias de la Computación, in: CACIC '18, 2018, pp. 452–461, http://sedici.unlp.edu.ar/handle/10915/73223.

[16] A. Alabdulatif, Privacy-Preserving Data Analytics in Cloud Computing (Ph.D. thesis), RMIT University, Melbourne, Australia, 2018, https://researchbank.rmit.edu.au/view/rmit:162567.

[17] A. Alabdulatif, I. Khalil, X. Yi, M. Guizani, Secure edge of things for smart healthcare surveillance framework, IEEE Access 7 (2019) 31010–31021, http://dx.doi.org/10.1109/ACCESS.2019.2899323.

[18] A. Alabdulatif, I. Khalil, X. Yi, Towards secure big data analytic for cloud-enabled applications with fully homomorphic encryption, J. Parallel Distrib. Comput. 137 (2020) 192–204, http://dx.doi.org/10.1016/j.jpdc.2019.10.008, http://www.sciencedirect.com/science/article/pii/S0743731519300887.

[19] P. Hao, E. Manhando, T. Ye, C. Bai, Video summarization based on sparse subspace clustering with automatically estimated number of clusters, in: Proceedings of the ACM Multimedia Asia, in: MMAsia '19, ACM, New York, NY, USA, 2019, http://dx.doi.org/10.1145/3338533.3366593.

[20] P. Olukanmi, F. Nelwamondo, T. Marwala, Rethinking *k*-means clustering in the age of massive datasets: A constant-time approach, Neural. Comput. Appl. (2019) 1–23, http://dx.doi.org/10.1007/s00521-019-04673-0.

[21] P. Olukanmi, F. Nelwamondo, T. Marwala, *k*-Means-Lite++: The combined advantage of sampling and seeding, in: 2019 6th International Conference on Soft Computing & Machine Intelligence, in: ISCMI '19, 2019, pp. 223–227, http://dx.doi.org/10.1109/ISCMI47871.2019.9004300.

[22] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., SciPy 1.0: Fundamental algorithms for scientific computing in Python, Nature Methods 17 (3) (2020) 261–272, http://dx.doi.org/10.1038/s41592-019-0686-2.

[23] S.K. Popuri, M.K. Gobbert, A Comparative Evaluation of Matlab, Octave, R, and Julia on Maya, Tech. Rep., (HPCF-2017-3) UMBC College of Natural and Mathematical Sciences, UMBC Faculty Collection, 2017, http://hdl.handle.net/11603/11302.