




Clustergram: Visualization and diagnostics for cluster analysis

Martin Fleischmann ¹

¹ Department of Social Geography and Regional Development, Charles University

DOI: [10.21105/joss.05240](https://doi.org/10.21105/joss.05240)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Charlotte Soneson](#)  

Reviewers:

- [@csadorf](#)
- [@gagolews](#)

Submitted: 16 January 2023

Published: 02 September 2023

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

Given a heterogeneous group of observations, researchers often try to find more homogenous groups within them. Typical is the use of clustering algorithms determining these groups based on statistical similarity. While there is an extensive range of algorithms to be chosen from, they often share one specific limitation - the algorithm itself will not determine the optimal number of clusters a group of observations shall be divided into. The solution is usually depending on internal cluster validity measures, but those provide only limited insight and can result in a suboptimal choice ([Gagolewski et al., 2021](#)). This paper presents a Python package named `clustergram` offering tools to analyze the clustering solutions and visualize the behavior of observations in relation to a tested range of options for the number of classes, enabling a deeper understanding of the behavior of observations splitting into classes and better-informed decisions on the optimal number of classes.

The situation the package is dealing with can be illustrated on one of the most commonly used clustering algorithms, K-Means. The algorithm first sets a pre-defined number of random seeds and attempts to split the data into the same number of classes, searching for the optimal seed locations providing the best split between the groups. However, the number of classes needs to be defined by a researcher and is usually unknown. The clustering solution is therefore created for a range of viable solutions (usually from 2 to N) that are compared and assessed based on various criteria, be it a so-called “*elbow plot*” of silhouette score looking for the “elbow” on a curve or a related silhouette analysis, or using other evaluation metrics, with both former and latter options often resulting in partitions that may not be relevant ([Gagolewski et al., 2021](#)). Most of them have in common that they treat each clustering option separately, without a relation between, e.g., when testing 3 and 4 clusters, the behavior of observations between these two options is not considered. To alleviate the situation and shed more light on the dynamics of *reshuffling* of observations between clusters, Schonlau ([2002](#)) proposed a new visual method called “*clustergram*”.

Clustergrams take the shape of a hierarchical diagram displaying a range of clustering options (number of clusters) on (usually) the X-axis and cluster centers for each solution on the Y-axis. Furthermore, there is an indication of a number of observations shifting between clusters, so we can see how large a portion of cluster A from a 2-cluster solution goes to cluster B of a 3-cluster solution, for example. This visualization uncovers the hierarchical nature of range-based series of clustering solutions and enables researchers to determine the optimal number of classes based on the illustrated behavior of observations as shown in [Figure 1](#), and [Figure 2](#) further explained below.

The Python package presented in this paper provides tools to create and explore clustergrams in Python based on a number of built-in clustering algorithms but also on external input resulting from other algorithms. The API is organized around a single overarching `clustergram.Clustergram` class designed around scikit-learn's API style ([Buitinck et al., 2013](#)) with initialization of the class with the specification of arguments and the `fit` method, making

it familiar to existing users of scikit-learn (Pedregosa et al., 2011) and similarly-designed packages. In its core, the class expects a selection of a range of solutions to be tested (`k_range`) from 1 to N , a selection of clustering algorithm (`method`) and a specification of a backend used for computation. Here, `clustergram` offers a choice between backends written to run on a CPU (scikit-learn for K-Means, Mini-batch K-Means and Gaussian Mixture Models, `scipy` (Virtanen et al., 2020) for hierarchical (or agglomerative) algorithms) or a GPU (`cuML` (Raschka et al., 2020) for K-Means), where the GPU path is computing both clustering and the underlying data for clustergram visualization on GPU, minimizing the need of data transfer between both. Furthermore, suppose none of the built-in options is suited for a set use case. In that case, the clustergram data structure can be created either from original data and labels for individual cluster solutions (`from_data()` method) or from cluster centers (`from_centers()` method), depending on the information obtainable from the selected external clustering algorithm.

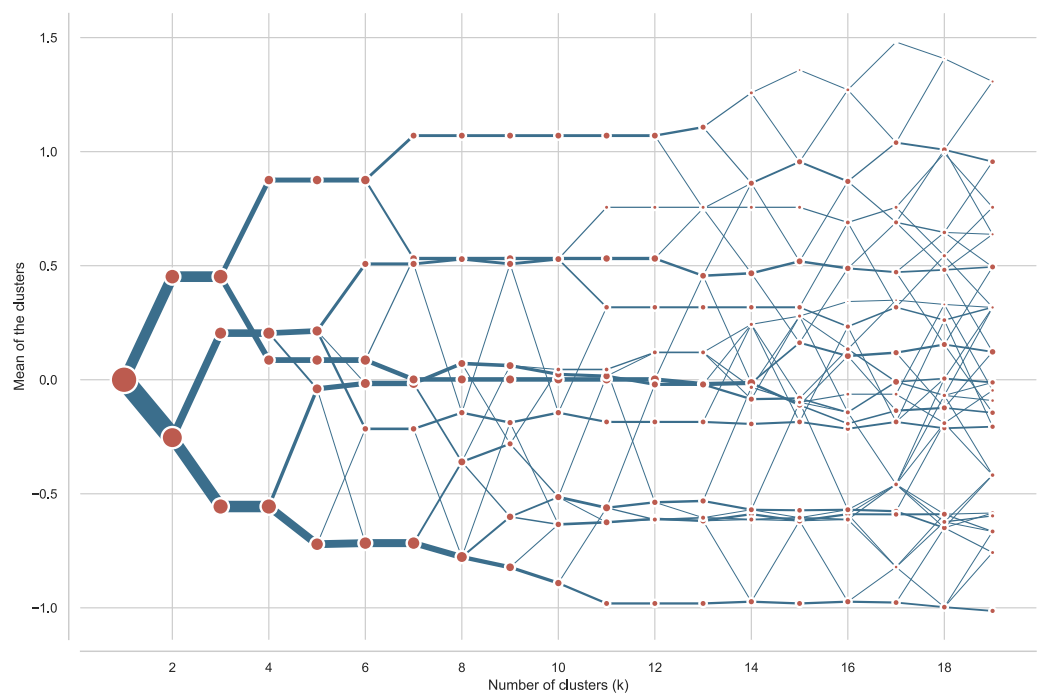


Figure 1: Clustergram based on the K-Means clustering algorithm as implemented in the scikit-learn package based on Palmer penguins dataset (Horst et al. (2020)). The cluster centroids are showing the non-weighted mean values as proposed in the original paper by Schonlau (2002).

Once the series of cluster solutions is generated, it is time to compute and generate clustergram diagrams for plotting functionality. The package offers two different ways of computing clustergram values. The first case shown in Figure 1 follows the original proposal by Schonlau (2002) and uses the means of cluster centroids to plot on Y-axis. However, as later noted by Galili (2010), using simple means does not necessarily result in the most readable clustergram. Therefore, the default option is to use the means of cluster centroids weighted by the first principal component derived from the complete dataset, shown in Figure 2 based on the same set of clustering solutions. Moreover, weighting by any other principal component is also available if a researcher needs further exploration. Due to the potential high computation cost of principal components and weighted cluster centroids, all the values are cached once computed, meaning that only the first plotting call triggers the computation.

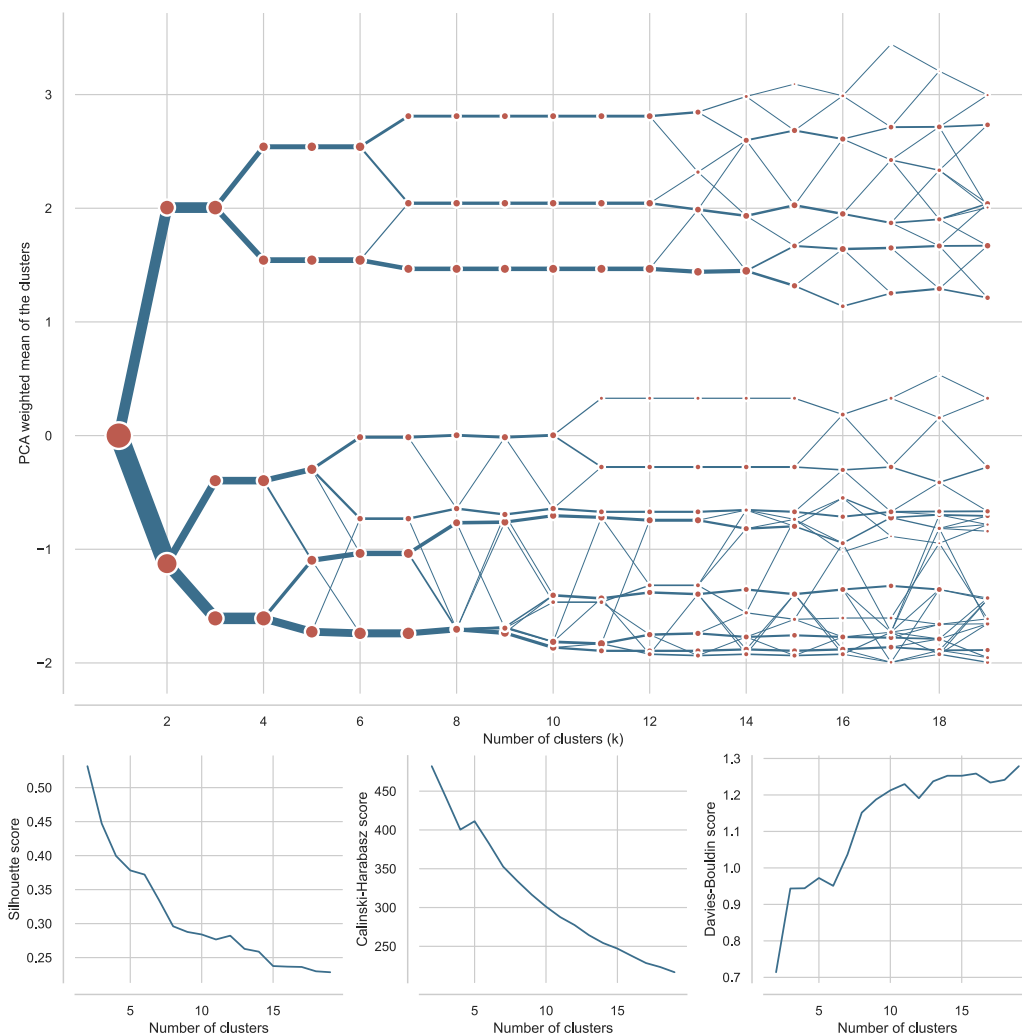


Figure 2: Clustergram based on the K-Means clustering algorithm as implemented in the scikit-learn package based on Palmer penguins dataset (Horst et al. (2020)), together with the additional metrics of cluster fit generated by the package. The cluster centroids are weighted by the first principal component to enhance the distinction between the branches of the dendrogram.

The plotting is implemented in two different options, showing the same diagram but one as a static matplotlib (Hunter, 2007) figure while the other as an interactive JavaScript-based visualization based on bokeh (Bokeh Team, 2023). The latter is especially beneficial as it offers direct links of cluster centroids within the diagram to individual labels allowing very granular back-and-forth diagnostics of the clustering behavior.

Since the selection of an optimal number of classes is a non-trivial exercise and shall not, in the ideal case, be left to a single method or metric, clustergram natively allows computation of additional metrics of cluster fit (Silhouette score, Calinski-Harabasz score, Davies-Bouldin score) directly from the main class using the implementation available in the scikit-learn, while the direct access to the labels resulting from all clustering options allows easy computation of any other similar metric.

Statement of need

As the problem `clustergram` helps resolve is not closed, there is a need for additional methods beyond the elbow plot and other traditionally used ways. It is clearly indicated by the constant citation level of the original set of papers by Schonlau (Schonlau, 2002, 2004). Arguably, this has been limited by the lack of ready-to-use implementation of the technique in the modern data science pipelines as the Schonlau (2002)'s code has been written in 2002 for STATA. Another implementation has been explored in a blog post by Galili (2010) experimenting with the minimal (as well as unpackaged and unmaintained) R version, that was later incorporated in the `EcotoneFinder` package (Bagnaro, 2021). Since the first release of `clustergram` in November 2020, the package has been used in at least seven academic publications, ranging from the classification of geographical areas based on form and function (Arribas-Bel & Fleischmann, 2022; Fleischmann & Arribas-Bel, 2022; Samardzhiev et al., 2022), geodemographics (Yang et al., 2022), clustering of the latent representation from convolutional neural networks (Singleton et al., 2022), classification of high Arctic lakes (Urbański, 2022) to facility reliability assessment (Stewart et al., 2022) and genomic data science (Ma et al., 2022). Since none of these directly cite the software, it is likely an incomplete overview. While researchers can still use the traditional set of metrics to estimate the optimal number of classes, none, including `clustergram`, is the ultimate answer without any drawbacks. What makes `clustergram` unique is the reflection of the dynamics of the sequence of solutions and the visualization of the behavior of observations within it.

Acknowledgements

The author kindly acknowledges the funding of the initial development by the UK's Economic and Social Research Council through the project "Learning an urban grammar from satellite data through AI", project reference ES/ T005238/1. Further appreciation belongs to Tal Galili, who popularized the method in the R world in 2010 and from whom `clustergram` borrowed its subtitle, *Visualization and diagnostics for cluster analysis*.

References

- Arribas-Bel, D., & Fleischmann, M. (2022). Spatial Signatures - Understanding (urban) spaces through form and function. *Habitat International*, 128, 102641. <https://doi.org/10.1016/j.habitatint.2022.102641>
- Bagnaro, A. (2021). *EcotoneFinder: Characterising and locating ecotones and communities*. <https://cran.r-project.org/web/packages/EcotoneFinder/index.html>
- Bokeh Team. (2023). *Bokeh/bokeh: Interactive Data Visualization in the browser, from Python*.
- Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: Experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.
- Fleischmann, M., & Arribas-Bel, D. (2022). Geographical characterisation of British urban form and function using the spatial signatures framework. *Scientific Data*, 9(1), 546. <https://doi.org/10.1038/s41597-022-01640-8>
- Gagolewski, M., Bartoszuk, M., & Cena, A. (2021). Are cluster validity measures (in) valid? *Information Sciences*, 581, 620–636. <https://doi.org/10.1016/j.ins.2021.10.004>

- Galili, T. (2010). *Clustergram: Visualization and diagnostics for cluster analysis (R code) | R-statistics blog*.
- Horst, A. M., Hill, A. P., & Gorman, K. B. (2020). *Palmerpenguins: Palmer archipelago (antarctica) penguin data*. <https://doi.org/10.5281/zenodo.3960218>
- Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- Ma, L., Peterson, E., Steliga, M., Muesse, J., Marino, K., Arnaoutakis, K., Shin, I., & Johann, D. J. (2022). Abstract 5038: Applying reproducible genomic data science methods for the analysis of a rare tumor type. *Cancer Research*, 82(12_Supplement), 5038. <https://doi.org/10.1158/1538-7445.AM2022-5038>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, 11(4). <https://doi.org/10.3390/info11040193>
- Samardzhiev, K., Fleischmann, M., Arribas-Bel, D., Calafiore, A., & Rowe, F. (2022). Functional signatures in Great Britain: A dataset. *Data in Brief*, 43, 108335. <https://doi.org/10.1016/j.dib.2022.108335>
- Schonlau, M. (2002). The clustergram: A graph for visualizing hierarchical and nonhierarchical cluster analyses. *The Stata Journal*, 2(4), 391–402. <https://doi.org/10.1177/1536867X0200200405>
- Schonlau, M. (2004). Visualizing non-hierarchical and hierarchical cluster analyses with clustergrams. *Computational Statistics*, 19(1), 95–111. <https://doi.org/10.1007/BF02915278>
- Singleton, A., Arribas-Bel, D., Murray, J., & Fleischmann, M. (2022). Estimating generalized measures of local neighbourhood context from multispectral satellite images using a convolutional neural network. *Computers, Environment and Urban Systems*, 95, 101802. <https://doi.org/10.1016/j.compenvurbsys.2022.101802>
- Stewart, R., Di Blasi, M., & Dessen, T. (2022, December). Addressing Data Gaps for Facility Reliability Assessments Using Non-Hierarchical Cluster Analysis. *2022 14th International Pipeline Conference*. <https://doi.org/10.1115/IPC2022-87145>
- Urbański, J. A. (2022). Monitoring and classification of high Arctic lakes in the Svalbard Islands using remote sensing. *International Journal of Applied Earth Observation and Geoinformation*, 112, 102911. <https://doi.org/10.1016/j.jag.2022.102911>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- Yang, Y., Dolega, L., & Darlington-Pollock, F. (2022). Ageing in Place Classification: Creating a geodemographic classification for the ageing population in England. *Applied Spatial Analysis and Policy*. <https://doi.org/10.1007/s12061-022-09490-y>