

# Monte Carlo Simulation of DBSCAN Parameter Effects on LiDAR Point Cloud

Mohamad Nweder

Department of Computer Engineering,  
University West, Trollhättan,  
Sweden ([m.neweder@gmail.com](mailto:m.neweder@gmail.com))

**Abstract**—This article presents a Monte Carlo simulation-based study on how DBSCAN parameters *epsilon* and *min\_samples* affect LiDAR point-cloud clustering. The simulation was performed by randomly varying these parameters over 2000 iterations, resulting in cluster counts ranging from X-Y and measuring cluster count, cluster size and noise ratio. The results demonstrate that DBSCAN is sensitive to parameter changes and that simulation can be a useful method to evaluate and analyze algorithm behavior without relying on real-world experiments.

**Index Terms**—*monte carlo, dbscan, lidar, clustering, simulation.*

## I. INTRODUCTION

Simulation methods are often used to understand important tools used today, as they save both money and time. Simulation is often applied to study algorithm behavior, control conditions and observe how algorithms respond to different parameters as input without requiring real-world experiments.

Methods used in simulation are applied in research to study and observe algorithm behavior under complex or uncertain conditions [1], [2]. In this experiment, a Monte Carlo Simulation was used to study the behavior of the DBSCAN clustering algorithm when its parameters change, using LiDAR point-cloud data obtained from PhD research work conducted at Luleå University of Technology (LTU) academic studies on DBSCAN under the supervision of Professor Ramin Karim [1]. DBSCAN algorithm is used to find clusters of varying density and has shown better performance than CLARNAS in benchmark tests [2]. The simulation executed DBSCAN repeatedly while randomly varying the parameters (*epsilon* and *min\_samples*) within predefined intervals. Each simulation run generated different values representing number of clusters, size of cluster the largest cluster and ratio of noise points.

Previous studies have discussed DBSCAN parameters sensitivity [2] and the use of Monte Carlo methods for algorithm evaluation [3]. The main goal of this simulation is to demonstrate how Monte Carlo simulation method can be used effectively to examine the sensitivity of changing parameters in DBSCAN algorithm without performing any physical or real-world experiments.

## II. METHODOLOGY

The simulation was developed to demonstrate how DBSCAN algorithm reacts when its parameters change repeatedly while running the code. Experiment was performed in Python using different libraries such as NumPy, Pandas.

### A. Simulation Steps

Simulation followed the Monte Carlo principle, which means that the same processes get repeated lot of times with different randomized data.

Every iteration was executed by DBSCAN algorithm with random parameters value:

- Epsilon ( $\epsilon$ ): Radius defines how close points must be to be considered neighbors.
- *min\_samples*: Minimum number of core points required to form a cluster.

Both parameters randomized between  $\epsilon = 0.05 - 0.25$  and *min\_samples* = 3 – 7. These parameters are chosen based on typical values used in previous A Density-Based Algorithm for Discovering Clusters experiments [2] and on the heuristic method described in section 4.2 of Ester et al. (1996).

Each running saved three results :

- 1- Number of founded cluster.
- 2- Largest cluster size.
- 3- Point parts that classed as cluster

### B. Material of data

The data used in this simulation originated from LiDAR point-cloud data conducted as PhD research at Luleå University of Technology (LTU) under the supervision of Professor Ramin Karim [1].

The dataset was available to students participating in a course project connected to doctoral research. it includes spatial coordinates from laser scanning of equipment surfaces and has been used in several studies related to Industrial AI and predictive maintenance.

Each datapoint had three coordinates (X,Y,Z). Ground level is calculated with a histogram that shows over Z-values to filter underground points. After filtering, the experiment will used only the above ground points.

### C. Implementation

A total 2000 iteration were executed using DBSCAN algorithm with different random parameters values. A fixed random seed was ensure reproducibility across all iterations. The simulation was implemented in Python using NumPy, Pandas, Matplotlib, and scikit-learn, DBSCAN and NearestNeighbors modules with standard modules os, sys, and Json for data handling and management.

The output from each iteration were automatically save for analysis, including overall static and summery data. Figures were generated showing histogram, cumulative distribution, convergent and sensitive measurement for all runed simulations. Data saved in CSV and JSON format are available in the appendix.

## III. SIMULATION RESULT

Simulation runs with totally 2000 iterations with help of DBSCAN algorithm. Parameters *epsilon* and *min\_samples* was randomized between given intervals. Each iteration saved the result locally in two different format, JSON and CSV. It gives the experiments opportunity to verify all runs and tests correctly and to present them as figures.

Two main figure will be purge down will shows exemplify results of simulation.

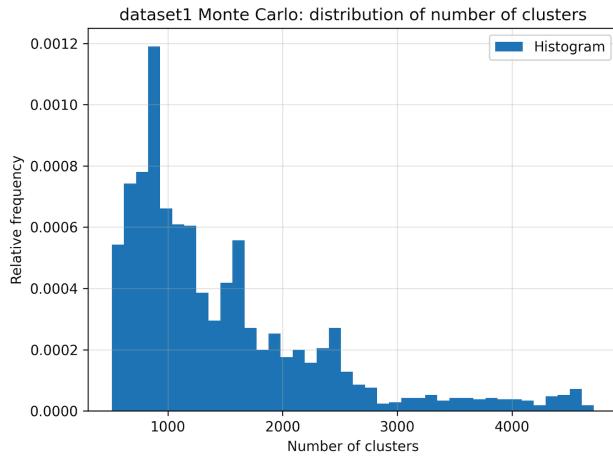


Fig. 1 Histogram of cluster counts.

Histogram in figure 1 shows how many cluster that founded between different iterations.

As shown in Fig. 1, X-axis present the numbers of clusters founded with the help of DBSCAN in each run. Every bar corresponds to an interval of cluster counts. Those values come from recorded results of each simulation run, where the number of clusters found by DBSCAN was saved automatically for every iteration.

The Y-axis shows how many iterations produced a specific range of cluster counts. If the bar was zero, then we don't have any cluster founded. The higher bar we have, the more iterations resulted in that range of cluster numbers.

Distribution stretches from the lowest value (fewer cluster) to higher values (many small cluster), which ensures that the random variations values of parameters worked correctly and that simulation performed as planned. Histogram also shows that most of runs resulted in values within the middle range of the interval, which means that our simulation generated stable and realistic spreads of the results.

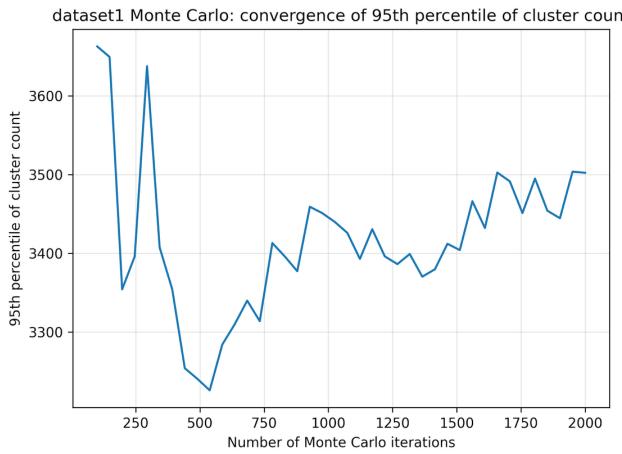


Fig 2 Convergence of cluster count (95th percentile).

As shown in Fig. 2 explain how the number of clusters (calculated as the 95th percentile) changes and stabilizes as the number of Monto Carlo iterations increases.

X-axis shows 95th percentile of the number of cluster found at each step, calculated progressively as the simulation proceeds.

It also shows that the result become stable after many iterations, which proves that the Carlo simulation has reached convergence. This confirms that the model works as expected and that's around on 2000 iterations to see stable result.

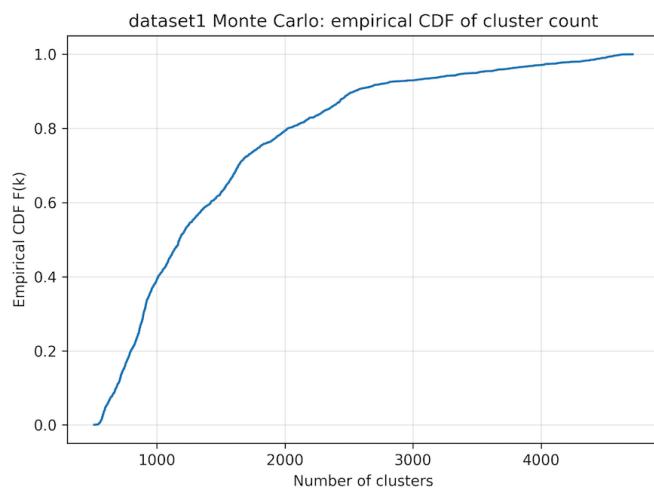


Fig. 3. Empirical cumulative distribution (ECDF) of cluster count.

As shown in Fig. 3 prove that the ECDF shows the cumulative probability that the number of clusters is below a given value, indicating a stable and continuous distribution across all Monte Carlo runs.

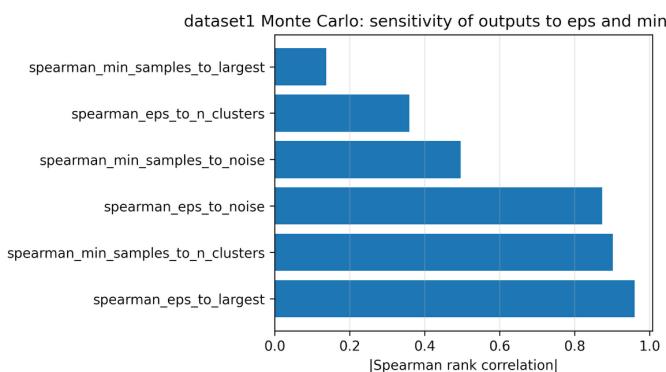


Fig. 4. Sensitivity analysis using Spearman correlation

As fig. 4 prove that how that  $\epsilon$ (epsilon) has the strongest influence on the number of clusters, confirming it as the dominant DBSCAN parameter.

#### IV. DISCUSSION

The simulation proved how useful the Monte Carlo method can be for investigating the sensitivity of parameter changes in DBSCAN without performing any real physical experiments.

The results also proved that our algorithm reacts strongly when we used and changed different values in  $\epsilon$  (epsilon) and even  $\text{min\_samples}$ . That means that our parameters have a big impact on how many clusters could be formed and how large the clusters are. We also saw that when we made 2000 different runs, it proved to be stable and enough to produce reproducible outcomes.

That proves that our developed simulation could be used for similar experiments but with different data or different algorithms to test how sensitive it could be when changing different parameters.

In the future, this simulation could be developed further, maybe by using another algorithm to compare this experiment with the new one and see what the differences would be, as well as to gain a better understanding of how sensitive it could be under different scenarios.

#### V. CONCLUSION

The simulation was done using the Monte Carlo method and showed that the simulation worked to make repeating runs and to investigate changes that happen when we change our parameters in a controlled and structured way.

By making 2000 different runs, it showed stability and reproducibility, proving that the simulation worked as expected and planned.

The experiment also showed that we can use this type of simulation in the future to reach results faster without physical testing, and it could be used for other studies with different datasets or algorithms.

#### REFERENCES

- [1] Luleå University of Technology, *LiDAR point-cloud data from PhD research supervised by Prof. Ramin Karim*, Luleå, Sweden, 2025.
- [2] J. Ester, M. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in Proc. KDD, 1996, pp. 226–231. [Online]. Available : <https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf>
- [3] N. Metropolis and S. Ulam, “The Monte Carlo method,” Journal of the American Statistical Association, vol. 44, no. 247, pp. 335–341, 1949. [Online]. Available: [https://web.williams.edu/Mathematics/sjmiller/public\\_html/105Sp10/handouts/MetropolisUlam\\_TheMonteCarloMethod.pdf](https://web.williams.edu/Mathematics/sjmiller/public_html/105Sp10/handouts/MetropolisUlam_TheMonteCarloMethod.pdf).

GITHUB

<https://github.com/Nweder/DTA400>