Dedenser: a Python command line tool for clustering and downsampling chemical libraries

Armen G. Beck[1*], Jonathan Fine[1,#], Yu-hong Lam[2,#], Edward C. Sherer[1], Erik L. Regalado[1], Pankaj Aggarwal[1*]

[1]*Analytical Research & Development, MRL, Merck & Co., Inc., Rahway, NJ, 07065, USA*
[2]*Modeling and Informatics, MRL, Merck & Co., Inc., Rahway, NJ, 07065, USA*

[*] Corresponding authors: armen.beck@merck.com (A.G.B) & pankaj.aggarwal@merck.com (P.A.)

ABSTRACT

The screening of chemical libraries is an essential starting point in the drug discovery process. While some researchers desire a more thorough screening of drug targets against a narrower scope of molecules, it is not uncommon for diverse screening sets to be favored during early stages of drug discovery. However, a cost burden is associated with the screening of molecules, with potential drawbacks if particular areas of chemical space are needlessly over represented. To facilitate triaged sampling of chemical libraries and other collections of molecules, we have developed Dedenser, a tool for the downsampling of chemical clusters. Dedenser functions by reducing the membership of clusters within chemical point clouds while maintaining the initial topology, or distribution, in chemical space. Dedenser is a Python package that utilizes Hierarchical Density-Based Spatial Clustering of Applications with Noise to first identify clusters present in 3D chemical point clouds, and then downsamples by applying Poisson disk sampling to clusters based on either their volume or density in chemical space. A command line interface tool is available with Dedenser, which allows for generation of chemical point clouds, using Mordred for QSAR descriptor calculations and uniform manifold approximation and projection for 3D embedding, as well as visualization. We hope that Dedenser will serve the community by enabling quick access to reduced collections of molecules that are representative of larger sets, selecting even distributions of molecules within clusters rather than single representative molecules.

INTRODUCTION

The designing of chemical libraries is a major component of the drug discovery process, with high throughput screening of libraries enabling the identification of lead compounds for further investigation and development.[1,2] Not long after this paradigm of chemical library synthesis and *in vitro* screening was introduced, there was incorporation of computational methods for the generation of initial chemical libraries and *in silico* screening.[3] For the generation of chemical libraries, fragment based and combinatorial approaches have historically been employed, [1,2,4] whereas structure-based approaches have been used for initial computational screening.[5,6] While *in silico* approaches to the discovery processes have become institutional mainstays with various established workflows, researchers continue to pursue novel computational methods for optimizing the generation, screening, and work-up of chemical libraries.

Machine learning methods to aid with assessment and *de novo* generation of molecules and chemical libraries in the drug development process are now commonplace.[7–9] These machine learning methods allow researchers to generate libraries and rank compounds based on various chemical features of molecules ranging from distinct physical chemical properties,[10–12] their predicted efficacy,[13,14] toxicity,[15–17] synthetic accessibility,[18–21] to other targeted properties[22]. Ensuring an appropriate degree of chemical diversity is a desired feature when working up chemical libraries, resulting in the development of a plethora of cheminformatic tools for the analysis and synthesis of chemical libraries and their diversity.[23–29]

The visualization of libraries can provide intuitive means of assessment and is a powerful addition to the various metrics developed by computational chemists for quantitative assessment and comparison of chemical libraries.[30] By taking higher dimensional representations of molecules, often quantitative structure-activity relationship (QSAR) descriptors, and embedding them as 2D or 3D vectors, the chemical space of libraries can be visualized. Once embedded and visualized, researchers are able to interactively analyze their chemical libraries and perform tasks such as clustering and selecting representative molecules.[31,32] Numerous approaches to clustering have been applied to chemical libraries, exploring various ways of selecting compounds from clusters. Considering nearest neighbors to cluster centroids has been a common way of producing representative sets of compounds.[31–34]

In the field of computer graphics, 3D collections of points are referred to as *point clouds* such that chemical clusters will be referred to as point clouds throughout this discussion.[35] While various applications surrounding point clouds exist, they are primarily used for 3D modeling and visualization of real world objects and spaces. By processing and analyzing point cloud data, it is possible to recreate the geometry and appearance of complex objects with high fidelity. Computer scientists have developed and adapted various algorithms for tasks such as data merging,[36] segmentation,[37] filtering,[38] and visualizations[39] to accomplish many tasks surrounding the application of point clouds.

Although representation of chemical space have been investigated for decades,[40–42] the development of point cloud based applications have been somewhat limited in the chemical sciences beyond pharmacophoric point clouds.[43–45] Point clouds have been recently repurposed for

chemistry when used for representing electrostatic potential surfaces,[46,47] molecular coordinates,[48,49] and interaction sites[50]. However, the native point cloud structure of 3D embedded chemical libraries and point cloud sampling algorithms from the computer graphics field have not been leveraged for diversity centric sampling of libraries.

In order to facilitate the curation of diverse and unbiased downsampled subsets of chemical libraries, we developed Dedenser. Available as a Python package, Dedenser is built around a downsampling algorithm which provides users with the ability to systematically reduce the number of molecules in a chemical library represented as a (chemical) point cloud. The downsampling process employed preserves the overall 3D topology of the chemical point cloud, and consequently the chemical space occupied by the set of molecules. This downsampling algorithm is largely defined as a two phase process: 1) the identification of clusters and singleton molecules and 2) volume/density based downsampling of clusters. For ease of use we have developed a command line interface for Dedenser, requiring no coding experience to utilize. Herein we discuss the details surrounding the theory, implementation, and brief case studies of Dedenser. Overall Dedenser provides an alternative to methods that draw single representative compounds from clusters, but instead provides diversity by downsampling the cluster such that an evenly distributed set of compounds is generated within each cluster based on relative cluster size.

METHODS AND THEORY

In this section, we discuss fundamental components of Dedenser including Poisson disk downsampling, clustering with an emphasis on the employed Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) algorithm, and how they function within the overall formulation of the algorithm.

Density Based Clustering

Unlike point cloud centric/specific algorithms, clustering has been employed as a major component of cheminformatics and machine learning application surrounding chemistry, such as during library design,[51,52] data driven synthesis,[53,54] molecular dynamics simulations,[55,56] and mass spectrometry[57,58]. Dedenser utilizes HDBSCAN,[59] a variant of DBSCAN,[60] that employs hierarchical trees to define clusters within a given dataset. Briefly, the HDBSCAN algorithm is executed by: 1) calculating mutual reachability distances between data points based on their distances and local densities, 2) building of a minimum spanning tree from the mutual reachability distances, 3) converting the minimum spanning tree into a hierarchical tree, 4) condensing the hierarchical tree such that nodes are clusters of data points, and 5) extraction of clusters from the condensed tree.

Regardless of the specifics surrounding its formulation, the choice to use HDBSCAN for clustering in Dedenser was due to two major aspects. First, the extraction of clusters is a data-dependent process, often eliminating the need for users to fine tune parameters and the number of clusters. Second, datapoints not belonging to clusters are considered ‘noise’, allowing for our Dedenser algorithm to retain singletons in chemical space. The retention of chemical singletons is of particular interest when the aim of downsampling is to retain chemical diversity in part.

Poisson Disk Downsampling

Dedenser's ability to reduce chemical point cloud membership to a target value is built around the Point Cloud Utils package,[61] a Python interface to C++ functions. Dedenser utilizes the function `downsample_point_cloud_poisson_disk` to achieve blue noise distributions. These blue noise distributions are defined as being evenly distributed points with no bias toward a specific radius from some central point and therefore the points should be distributed independently and randomly (Figure 1). Poisson disk sampling is utilized to downsample to a blue noise distribution.[62]

The Poisson disk downsampling function (f) as implemented in Point Cloud Utils is an iterative algorithm which will output a blue noise distributed point cloud (P^*) when provided an initial point cloud (P) and a target number of points to keep (n). The Poisson disk represents a 3D disk in our case, though it generally may be N-dimensional, defined by a radius (r). Poisson disks are used to define a lower distance bound, ensuring that that no two points (p) are within a Poisson disk centered around any other point when (down)sampling. In our case, the distance (d) between points is the L_2 norm in 3D Euclidean space (Euclidean distance), with the formalization of point cloud downsampling via Poisson disks displayed in eq. 1:

$$(1)\ P^* := f(P) = \{p_1, \dots p_n\}, d(p_i, p_j) \geq r, \forall p_i, p_j \in P^*$$

When provided a target number of points, the Poisson disk downsampling of point clouds is initialized by generating grid cells within the spatial boundaries of the point cloud and calculating upper and lower sample size tolerances. Since the radius is not provided, in our case, it is first defined in relation to the spatial size of the point cloud, and then adaptively scaled as the algorithm iterates. During each iteration, grid cells are generated with face lengths (s), where $s = r/\sqrt{3}$, ensuring that each grid cell can only contain a single point given the Poisson disk constraint. Given the radius resulting grid cells, a random grid cell with points inside is selected to initiate the inner-loop of the algorithm and the downsampling of the point cloud. After the initial grid cell is selected, a random point within the cell is selected followed by the selection of an adjacent cell which in turn has its points checked for one being outside the Poisson disks of other selected points. The process then continues until all grid cells have been evaluated, and is repeated with varying radius values until the downsampled point cloud falls within the target size tolerances.

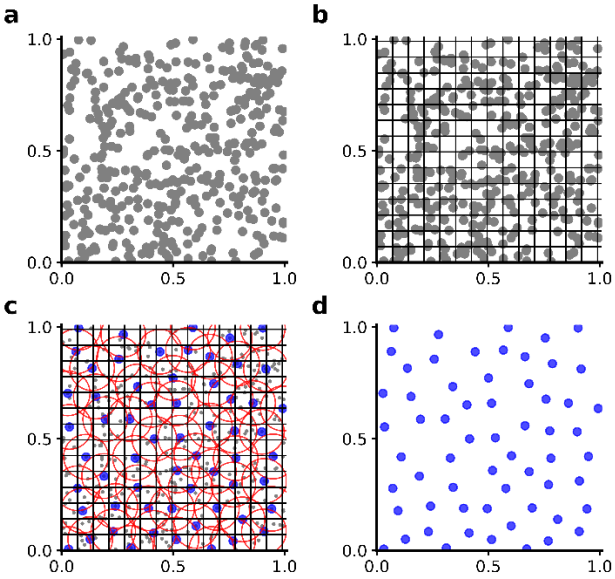

Figure 1. Overview of Poisson disk downsampling. (a) Uniform distribution of points generated by random sampling. (b) Grid with face lengths of s. (c) Selection of points for downsampling (blue) with no more than one point per grid-cell nor within the Poisson disk of $r = s\sqrt{2}$ centered around each selected point. (d) Blue noise distribution achieved by Poisson disk downsampling.

Dedenser Algorithm

The Dedenser algorithm is formulated to first cluster a chemical point cloud with HDBSCAN and then apply Poisson disk downsampling to clusters based on their relative hull volumes or weighted densities. The goal of downsampling is to achieve a targeted percentage of retained compounds from the initial chemical point cloud. To calculate the number of molecules each cluster is to be downsampled to a remaining target (T*) is calculated as the difference between the initial target (T) and the number of chemical singletons (S), or 'noise' (eq 2.). The hull volume for each cluster (v_i) is then calculated, and used to determine the relative volume of each cluster which is then used to calculate the target number of molecules for each cluster (T_i^*) (eq 3.). Alternatively, Dedenser allows for cluster targets to be calculated from proportional weightings (W_i) of clusters (eq. 4), derived from exponential functions parameterized by a weighting term (w) to allow for greater differentiation between cluster properties (p_i, either cluster density or volume) (eq. 5). Additionally, by assigning negative weights, sparser/smaller clusters will be assigned greater relative target sizes compared to denser/larger clusters.

(2) $T^* = T - S$

(3) $T_i^* = T^* \times v_i / \sum_{i=1}^{n} v_i \, , n = number\ of\ clusters$

(4) $T_i^* = T^* \times W_i$

(5) $W_i = e^{w(p_i/p_T - 1)} / \sum_{i=1}^{n} e^{w(p_i/p_T - 1)}, \; p_T = \sum_{i=1}^{n} p_i$

The two major caveats to the algorithm functioning as described above are greater noise membership than the overall target value of downsampling and cluster targets being greater than their overall membership. For the former case, where the number of chemical singletons is initially greater than the desired number of molecules to be retained, Dedenser applies Poisson disk

downsampling to the noise before downsampling clusters. In this case, the target for noise downsampling is defined as 80% of its equivalent, volume based target, effectively treating it as a cluster in eq. 2. For the latter case, where clusters may have membership lower than their respective target for downsampling, an iterative loop is executed that retains all the molecules in these below target clusters, and subsequently recalculates the remaining targets. This iterative process of checking and managing the cluster targets is executed until all remaining clusters have initial memberships greater than their respective target membership for downsampling. Further details regarding the two-stage downsampling algorithm and some implementation details defining Dedenser are outlined in pseudocode (Algorithm S1.) for Dedenser using volume based cluster downsampling.

One key advantage to Dedenser's cluster based downsampling over simply applying Poisson disk downsampling to the entire chemical point cloud is the potential to capture all chemical singletons rather than simply reducing them to blue noise. Additionally, dense clusters would be represented proportionally, by volume or density in chemical space, rather than also being reduced to blue noise relatively with the rest of the chemical point cloud. This may be critical when downsampling libraries, such as those generated in a targeted manner. Per example, if one were to generate an initial chemical library via a Bayesian sampling process[10], areas with higher likelihoods from the fitted prior distribution may be desired for retention at a higher proportionality, yet still downsampled. Alternatively, if desired, the epsilon parameter for cluster selection in HDBSCAN can be increased to treat all data as one cluster, effectively performing Poisson disk downsampling on the entire chemical point cloud.

IMPLEMENTATION

Dedenser is implemented such that it can be run as a command line tool, or as a Python library. Additionally, while it contains built-in functions for chemical descriptor calculation and embedding to 3D chemical space, the novel functionality regarding downsampling is agnostic and can handle any 3D point cloud representation.

When developing Dedenser, we utilized Python 3 and various packages/libraries to enable the functionalities surrounding the Dedenser algorithm and user centric utilities. Regarding the implementation of the Dedenser algorithm various libraries were used as follows: scikit-learn[63] for clustering with HDBSCAN and Gaussian Mixture Models, SciPy[64] for convex hull calculations, Alphashape[65] for concave hull calculations, and Point Cloud Utils[61] for Poisson-disk downsampling. Utility functions from the following libraries were used: pandas[66] for data handling reading/writing of files, openpyxl[67] for extending pandas read/write features to Excel, scikit-learn[63] for standard scalar calculations, Matplotlib for visualization, RDKit[68] and Mordred[69] for chemical descriptor calculations, and UMAP[70] for embedding descriptor into chemical point clouds. Additionally, NumPy[71] was used for various data manipulations and reading/writing operations, with all further dependencies captured in the requirements.txt and ddenser.yml files hosted on GitHub.

Dedenser Library

As a library, Dedenser is comprised of the main Dedenser class and utility functions for generating, visualizing and saving the indexes of chemical point clouds. Technical aspects and formulation of Dedenser and the chemical point cloud utilities are explained in detail within the Dedenser documentation at https://github.com/MSDLLCpapers/dedenser. We would hope that others in the field would feel encouraged to incorporate their desired variations to chemical point cloud representations or modify the downsampling algorithm of Dedenser as they see fit, and that this may be assisted through documentation of the codebase. Additionally, a general workflow and implementation details are expressed in the supporting information. Regarding HDBSCAN, only epsilon and minimum cluster size are parameterized.

CASE STUDIES

To capture the usage and behavior of Dedenser with tangible examples of application, we have performed case studies with two datasets. The first, involves a subset of the ZINC[72] compounds extracted by Gómez-Bombarelli et al.,[73] the second is a set of aryl bromides from a study reported by Kariofillis et al.[31].

Downsampling of ZINC Compounds

A 1000 member subset of the initial ZINC compound database was generated by taking nearest neighbors from a multivariate bimodal distribution among logP, molecular refractivity, molecular weight, polarizable surface area, and the complexity index BertzCT descriptors calculated using Mordred. Once descriptors were calculated, they were embedded into a 3D chemical point cloud to allow for downsampling.

Using default parameters, two clusters were identified in the initial chemical point cloud of 1000 compounds and was downsampled with a target membership of 10%, or 100 molecules, of the initial set and yielded a downsampled set of 99 molecules. The overall topology of the initial chemical point cloud (Figure 2A) is preserved post downsampling (Figure 2B) when using the default Dedenser parameters. The original distributions of the ZINC compounds was largely retained when downsampled (Figure 2C), demonstrating Dedenser's ability to preserve chemical diversity when reducing chemical point cloud membership.

To showcase the ability of Dedenser to leverage weighted cluster downsampling, positive and negative weighting of cluster densities were employed. When using a positive density weighting of 5, the cluster occupying the lower range of UMAP dimensions 0 and 2 is given greater membership (Figure S1). Conversely, by using a weighting of -5, the alternative cluster has greater membership (Figure S2). For the negative and positive weightings, molecules with lower and greater descriptor values are favored respectively and are distinct from random sampling (Figure S3). In both cases, the downsampling done with Dedenser shows fewer clumping of molecules and more regular distribution in embedded chemical descriptor space when compared to random sampling (Figures S1-2).

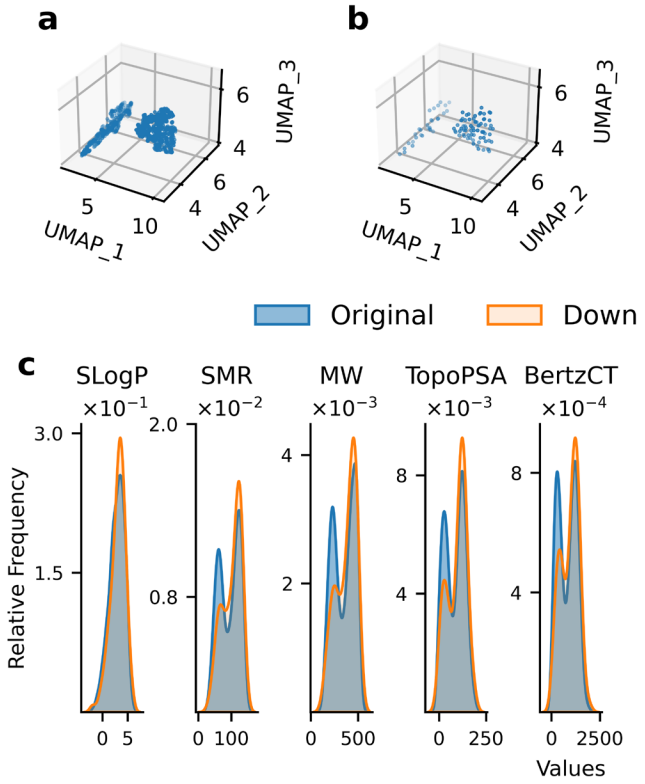

Figure 2. Chemical point cloud generated and downsampled ZINC compounds using Dedenser with their descriptor distributions. (a) Initial chemical point cloud composed of 1000 compounds from ZINC by projecting their Mordred descriptors with UMAP. (b) Downsampled chemical point cloud with a membership of 99 molecules, utilizing default parameters while targeting 10% initial membership. (C) Density plots displaying the Mordred descriptor distributions of the original (blue) and downsampled (orange) distributions of Wildman-Crippen LogP (SLogP), Wildman-Crippen molar refractivity (SMR), molecular weight (MW), topological polarizable surface area (TopoPSA), and Bertz index (BertzCT).

Downsampling Aryl Bromides

To provide a more complex chemical space, we used the set of aryl bromides from Kariofillis et al.[31]. Instead of calculating only selected descriptors, all 2D Mordred descriptors were calculated for the aryl bromides. Again, once descriptors were calculated, they were embedded into a 3D chemical point cloud to allow for downsampling.

Using the default parameters for downsampling to a target of 25% membership (671 molecules), the initial population of 2683 was downsampled to 763 molecules. While the topography of the original chemical point cloud (Figures 3A & S4) and the downsampled point cloud (Figures 3B & S5) remain similar, and the distributions for selected descriptors is roughly scaled down (Figure 3C), a major discrepancy in the targeted membership from downsampling is apparent. A large portion of this discrepancy is due to 103 clusters having targeted memberships of zero, with one member being retained as the nearest neighbor to the cluster centroid. By using the visualization flag, a large amount of clusters can be noted in addition to many chemical singletons (Figures S4-5).

To promote larger clusters by HDBSAN and favor larger clusters by volume, we set the epsilon and volume weighting parameters to 0.22 and 4 respectively. By doing so, the membership post downsampling was decreased to 657 molecules, a 12% reduction in error due to elimination of miniscule clusters. The increase in cluster sizes is clearly illustrated in the UMAP plot generated using the visualization flag (Fig S6). Overall the command line functionalities and rapid visualization included in Dedenser allows for user friendly curation of downsampled chemical libraries.

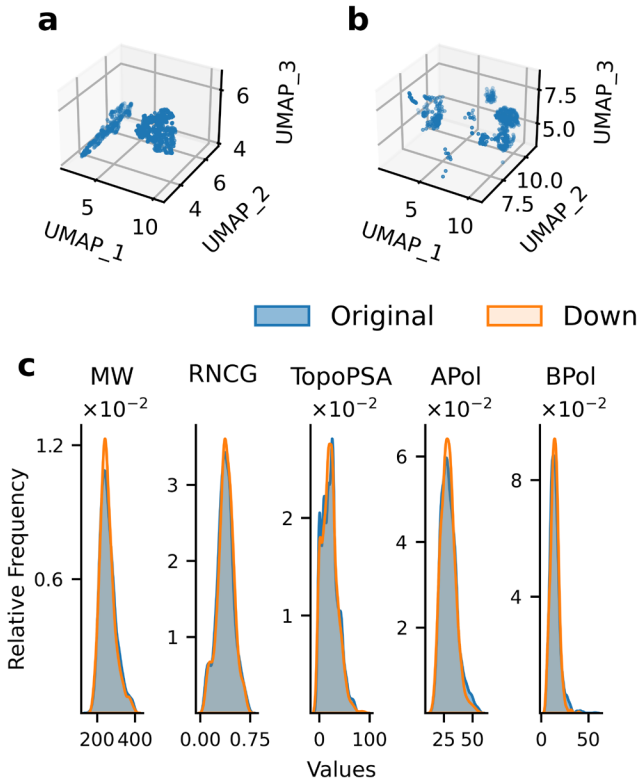

Figure 3. Chemical point cloud generated and downsampled aryl bromides using Dedenser. (a) Initial chemical point cloud composed of 2683 aryl bromides by projecting their Mordred descriptors with UMAP. (b) Downsampled chemical point cloud with a membership of 736 aryl bromides, utilizing default parameters while targeting 25% initial membership. (C) Density plots displaying the Mordred descriptor distributions of the original (blue) and downsampled (orange) distributions of molecular weight (MW), relative negative charge (RNCG), topological polar surface area (TopoPSA), atomic polarizability (APol), and bond polarizability (BPol).

CONCLUSIONS

Due to the overhead presented by the synthesis, procurement, and screening of chemical libraries, triaged approaches to formulating optimal libraries are desired to alleviate such resource constraints. To this end, we have developed Dedenser which functions by downsampling chemical point clouds to remove molecules from high density regions in chemical space, providing chemical libraries with reduced membership while maintaining diversity. We hope that researchers will find

utility in Dedenser, as it can facilitate the generation of uniformly distributed clusters in datasets drawn from irregularly populated chemical space. Additionally, a command line interface tool version of Dedenser allows for researchers to generate, downsample, and visualize chemical point clouds without any need to code. Beyond the explanation for usage described within this manuscript, Dedenser is accompanied by full documentation to assist those that wish to utilize their own chemical point clouds or further modify the code.

Author Contributions

[#]J.F. and YH.L. contributed equally to this work.

Conflicts of Interest

All authors are employees of Merck Sharp & Dohme LLC, a subsidiary of Merck & Co., Inc., Rahway, NJ, USA.

Funding

The authors would like to thank the MRL Postdoctoral Research Fellow Program for sponsoring this research. This work was fully funded by Merck Sharp & Dohme LLC, a subsidiary of Merck & Co., Inc., Rahway, NJ, USA.

Data Availability

The Python code for Dedenser and data from use cases are publicly available at https://github.com/MSDLLCpapers/dedenser.

REFERENCES

(1) Drews, J. Drug Discovery: A Historical Perspective. *Science (80-.).* **2000**, *287* (5460), 1960–1964. https://doi.org/10.1126/science.287.5460.1960.

(2) Dolle, R. E. Historical Overview of Chemical Library Design; 2011; pp 3–25. https://doi.org/10.1007/978-1-60761-931-4_1.

(3) Hopfinger, A. J. Computer-Assisted Drug Design. *J. Med. Chem.* **1985**, *28* (9), 1133–1139. https://doi.org/10.1021/jm00147a001.

(4) Gordon, E. M.; Barrett, R. W.; Dower, W. J.; Fodor, S. P. A.; Gallop, M. A. Applications of Combinatorial Technologies to Drug Discovery. 2. Combinatorial Organic Synthesis, Library Screening Strategies, and Future Directions. *J. Med. Chem.* **1994**, *37* (10), 1385–1401. https://doi.org/10.1021/jm00036a001.

(5) Ghosh, S.; Nie, A.; An, J.; Huang, Z. Structure-Based Virtual Screening of Chemical Libraries for Drug Discovery. *Curr. Opin. Chem. Biol.* **2006**, *10* (3), 194–202. https://doi.org/10.1016/j.cbpa.2006.04.002.

(6) van Dongen, M.; Weigelt, J.; Uppenberg, J.; Schultz, J.; Wikström, M. Structure-Based Screening and Design in Drug Discovery. *Drug Discov. Today* **2002**, *7* (8), 471–478. https://doi.org/10.1016/S1359-6446(02)02233-X.

(7) Panteleev, J.; Gao, H.; Jia, L. Recent Applications of Machine Learning in Medicinal Chemistry. *Bioorg. Med. Chem. Lett.* **2018**, *28* (17), 2807–2815. https://doi.org/10.1016/j.bmcl.2018.06.046.

(8) Lo, Y.-C.; Rensi, S. E.; Torng, W.; Altman, R. B. Machine Learning in Chemoinformatics and Drug Discovery. *Drug Discov. Today* **2018**, *23* (8), 1538–1546. https://doi.org/10.1016/j.drudis.2018.05.010.

(9) Pang, C.; Qiao, J.; Zeng, X.; Zou, Q.; Wei, L. Deep Generative Models in De Novo Drug Molecule Generation. *J. Chem. Inf. Model.* **2024**, *64* (7), 2174–2194. https://doi.org/10.1021/acs.jcim.3c01496.

(10) Ikebata, H.; Hongo, K.; Isomura, T.; Maezono, R.; Yoshida, R. Bayesian Molecular Design with a Chemical Language Model. *J. Comput. Aided. Mol. Des.* **2017**, *31* (4), 379–391. https://doi.org/10.1007/s10822-016-0008-z.

(11) Vasudevan, R. K.; Choudhary, K.; Mehta, A.; Smith, R.; Kusne, G.; Tavazza, F.; Vlcek, L.; Ziatdinov, M.; Kalinin, S. V.; Hattrick-Simpers, J. Materials Science in the Artificial Intelligence Age: High-Throughput Library Generation, Machine Learning, and a Pathway from Correlations to the Underpinning Physics. *MRS Commun.* **2019**, *9* (3), 821–838. https://doi.org/10.1557/mrc.2019.95.

(12) Popova, M.; Isayev, O.; Tropsha, A. Deep Reinforcement Learning for de Novo Drug Design. *Sci. Adv.* **2018**, *4* (7). https://doi.org/10.1126/sciadv.aap7885.

(13) Grechishnikova, D. Transformer Neural Network for Protein-Specific de Novo Drug Generation as a Machine Translation Problem. *Sci. Rep.* **2021**, *11* (1), 321. https://doi.org/10.1038/s41598-020-79682-4.

(14) Podlewska, S.; Czarnecki, W. M.; Kafel, R.; Bojarski, A. J. Creating the New from the Old: Combinatorial Libraries Generation with Machine-Learning-Based Compound Structure Optimization. *J. Chem. Inf. Model.* **2017**, *57* (2), 133–147. https://doi.org/10.1021/acs.jcim.6b00426.

(15) Vo, A. H.; Van Vleet, T. R.; Gupta, R. R.; Liguori, M. J.; Rao, M. S. An Overview of Machine Learning and Big Data for Drug Toxicity Evaluation. *Chem. Res. Toxicol.* **2020**, *33* (1), 20–37. https://doi.org/10.1021/acs.chemrestox.9b00227.

(16) Tang, W.; Chen, J.; Wang, Z.; Xie, H.; Hong, H. Deep Learning for Predicting Toxicity of Chemicals: A Mini Review. *J. Environ. Sci. Heal. Part C* **2018**, *36* (4), 252–271. https://doi.org/10.1080/10590501.2018.1537563.

(17) Lu, J.; Lu, D.; Fu, Z.; Zheng, M.; Luo, X. Machine Learning-Based Modeling of Drug Toxicity; 2018; pp 247–264. https://doi.org/10.1007/978-1-4939-7717-8_15.

(18) Podolyan, Y.; Walters, M. A.; Karypis, G. Assessing Synthetic Accessibility of Chemical Compounds Using Machine Learning Methods. *J. Chem. Inf. Model.* **2010**, *50* (6), 979–991. https://doi.org/10.1021/ci900301v.

(19) Voršilák, M.; Kolář, M.; Čmelo, I.; Svozil, D. SYBA: Bayesian Estimation of Synthetic Accessibility of Organic Compounds. *J. Cheminform.* **2020**, *12* (1), 35. https://doi.org/10.1186/s13321-020-00439-2.

(20) Thakkar, A.; Chadimová, V.; Bjerrum, E. J.; Engkvist, O.; Reymond, J.-L. Retrosynthetic Accessibility Score (RAscore) – Rapid Machine Learned Synthesizability Classification from AI Driven Retrosynthetic Planning. *Chem. Sci.* **2021**, *12* (9), 3339–3349.

https://doi.org/10.1039/D0SC05401A.

(21) Parrot, M.; Tajmouati, H.; da Silva, V. B. R.; Atwood, B. R.; Fourcade, R.; Gaston-Mathé, Y.; Do Huu, N.; Perron, Q. Integrating Synthetic Accessibility with AI-Based Generative Drug Design. *J. Cheminform.* **2023**, *15* (1), 83. https://doi.org/10.1186/s13321-023-00742-8.

(22) de Oliveira, E. C. L.; Santana, K.; Josino, L.; Lima e Lima, A. H.; de Souza de Sales Júnior, C. Predicting Cell-Penetrating Peptides Using Machine Learning Algorithms and Navigating in Their Chemical Space. *Sci. Rep.* **2021**, *11* (1), 7628. https://doi.org/10.1038/s41598-021-87134-w.

(23) González-Medina, M.; Prieto-Martínez, F. D.; Owen, J. R.; Medina-Franco, J. L. Consensus Diversity Plots: A Global Diversity Analysis of Chemical Libraries. *J. Cheminform.* **2016**, *8* (1), 63. https://doi.org/10.1186/s13321-016-0176-9.

(24) Dunn, T. B.; Seabra, G. M.; Kim, T. D.; Juárez-Mercado, K. E.; Li, C.; Medina-Franco, J. L.; Miranda-Quintana, R. A. Diversity and Chemical Library Networks of Large Data Sets. *J. Chem. Inf. Model.* **2022**, *62* (9), 2186–2201. https://doi.org/10.1021/acs.jcim.1c01013.

(25) Krier, M.; Bret, G.; Rognan, D. Assessing the Scaffold Diversity of Screening Libraries. *J. Chem. Inf. Model.* **2006**, *46* (2), 512–524. https://doi.org/10.1021/ci050352v.

(26) Lenci, E.; Trabocchi, A. Diversity-Oriented Synthesis and Chemoinformatics: A Fruitful Synergy towards Better Chemical Libraries. *European J. Org. Chem.* **2022**, *2022* (29). https://doi.org/10.1002/ejoc.202200575.

(27) Colliandre, L.; Le Guilloux, V.; Bourg, S.; Morin-Allory, L. Visual Characterization and Diversity Quantification of Chemical Libraries: 2. Analysis and Selection of Size-Independent, Subspace-Specific Diversity Indices. *J. Chem. Inf. Model.* **2012**, *52* (2), 327–342. https://doi.org/10.1021/ci200535y.

(28) Le Guilloux, V.; Colliandre, L.; Bourg, S.; Guénegou, G.; Dubois-Chevalier, J.; Morin-Allory, L. Visual Characterization and Diversity Quantification of Chemical Libraries: 1. Creation of Delimited Reference Chemical Subspaces. *J. Chem. Inf. Model.* **2011**, *51* (8), 1762–1774. https://doi.org/10.1021/ci200051r.

(29) O'Hagan, S.; Kell, D. B. Analysing and Navigating Natural Products Space for Generating Small, Diverse, But Representative Chemical Libraries. *Biotechnol. J.* **2018**, *13* (1). https://doi.org/10.1002/biot.201700503.

(30) Medina-Franco, J.; Martinez-Mayorga, K.; Giulianotti, M.; Houghten, R.; Pinilla, C. Visualization of the Chemical Space in Drug Discovery. *Curr. Comput. Aided-Drug Des.* **2008**, *4* (4), 322–333. https://doi.org/10.2174/157340908786786010.

(31) Kariofillis, S. K.; Jiang, S.; Żurański, A. M.; Gandhi, S. S.; Martinez Alvarado, J. I.; Doyle, A. G. Using Data Science To Guide Aryl Bromide Substrate Scope Analysis in a Ni/Photoredox-Catalyzed Cross-Coupling with Acetals as Alcohol-Derived Radical Sources. *J. Am. Chem. Soc.* **2022**, *144* (2), 1045–1055. https://doi.org/10.1021/jacs.1c12203.

(32) Rana, D.; Pflüger, P. M.; Hölter, N. P.; Tan, G.; Glorius, F. Standardizing Substrate Selection: A Strategy toward Unbiased Evaluation of Reaction Generality. *ACS Cent. Sci.* **2024**. https://doi.org/10.1021/acscentsci.3c01638.

(33) Menard, P. R.; Lewis, R. A.; Mason, J. S. Rational Screening Set Design and Compound Selection: Cascaded Clustering. *J. Chem. Inf. Comput. Sci.* **1998**, *38* (3), 497–505. https://doi.org/10.1021/ci980003j.

(34) Willett, P.; Winterman, V.; Bawden, D. Implementation of Nonhierarchic Cluster Analysis Methods in Chemical Information Systems: Selection of Compounds for Biological Testing and Clustering of Substructure Search Output. *J. Chem. Inf. Comput. Sci.* **1986**, *26* (3), 109–118. https://doi.org/10.1021/ci00051a005.

(35) Camuffo, E.; Mari, D.; Milani, S. Recent Advancements in Learning Algorithms for Point Clouds: An Updated Overview. *Sensors* **2022**, *22* (4), 1357. https://doi.org/10.3390/s22041357.

(36) Kwon, S.; Park, J.-W.; Moon, D.; Jung, S.; Park, H. Smart Merging Method for Hybrid Point Cloud Data Using UAV and LIDAR in Earthwork Construction. *Procedia Eng.* **2017**, *196*, 21–28. https://doi.org/10.1016/j.proeng.2017.07.168.

(37) Nguyen, A.; Le, B. 3D Point Cloud Segmentation: A Survey. In *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*; IEEE, 2013; pp 225–230. https://doi.org/10.1109/RAM.2013.6758588.

(38) Han, X.-F.; Jin, J. S.; Wang, M.-J.; Jiang, W.; Gao, L.; Xiao, L. A Review of Algorithms for Filtering the 3D Point Cloud. *Signal Process. Image Commun.* **2017**, *57*, 103–112. https://doi.org/10.1016/j.image.2017.05.009.

(39) Richter, R.; Döllner, J. Concepts and Techniques for Integration, Analysis and Visualization of Massive 3D Point Clouds. *Comput. Environ. Urban Syst.* **2014**, *45*, 114–124. https://doi.org/10.1016/j.compenvurbsys.2013.07.004.

(40) Lipinski, C.; Hopkins, A. Navigating Chemical Space for Biology and Medicine. *Nature* **2004**, *432* (7019), 855–861. https://doi.org/10.1038/nature03193.

(41) Reymond, J.-L. The Chemical Space Project. *Acc. Chem. Res.* **2015**, *48* (3), 722–730. https://doi.org/10.1021/ar500432k.

(42) Reymond, J. L.; Van Deursen, R.; Blum, L. C.; Ruddigkeit, L. Chemical Space as a Source for New Drugs. *Medchemcomm* **2010**, *1* (1), 30–38. https://doi.org/10.1039/c0md00020e.

(43) Wang, H.; Mulgaonkar, N.; Pérez, L. M.; Fernando, S. ELIXIR-A: An Interactive Visualization Tool for Multi-Target Pharmacophore Refinement. *ACS Omega* **2022**, *7* (15), 12707–12715. https://doi.org/10.1021/acsomega.1c07144.

(44) Eguida, M.; Rognan, D. A Computer Vision Approach to Align and Compare Protein Cavities: Application to Fragment-Based Drug Design. *J. Med. Chem.* **2020**, *63* (13), 7127–7142. https://doi.org/10.1021/acs.jmedchem.0c00422.

(45) Schaller, D.; Šribar, D.; Noonan, T.; Deng, L.; Nguyen, T. N.; Pach, S.; Machalz, D.;

Bermudez, M.; Wolber, G. Next Generation 3D Pharmacophore Modeling. *WIREs Comput. Mol. Sci.* **2020**, *10* (4). https://doi.org/10.1002/wcms.1468.

(46) Wang, L.; Zhao, L.; Liu, X.; Fu, J.; Zhang, A. SepPCNET: Deeping Learning on a 3D Surface Electrostatic Potential Point Cloud for Enhanced Toxicity Classification and Its Application to Suspected Environmental Estrogens. *Environ. Sci. Technol.* **2021**, *55* (14), 9958–9967. https://doi.org/10.1021/acs.est.1c01228.

(47) Lv, K.; Zhang, J.; Liu, X.; Zhou, Y.; Liu, K. Computer-Aided Accurate Calculation of Interacted Volumes for 3D Isosurface Point Clouds of Molecular Electrostatic Potential. *J. Mol. Graph. Model.* **2024**, *126*, 108648. https://doi.org/10.1016/j.jmgm.2023.108648.

(48) Wu, Y.; Wei, H.; Zhu, Q.; Luo, R. Grid-Robust Efficient Neural Interface Model for Universal Molecule Surface Construction from Point Clouds. *J. Phys. Chem. Lett.* **2023**, *14* (40), 9034–9041. https://doi.org/10.1021/acs.jpclett.3c02176.

(49) Wang, Y.; Wu, S.; Duan, Y.; Huang, Y. A Point Cloud-Based Deep Learning Strategy for Protein–Ligand Binding Affinity Prediction. *Brief. Bioinform.* **2022**, *23* (1). https://doi.org/10.1093/bib/bbab474.

(50) Yan, X.; Lu, Y.; Li, Z.; Wei, Q.; Gao, X.; Wang, S.; Wu, S.; Cui, S. PointSite: A Point Cloud Segmentation Tool for Identification of Protein Ligand Binding Atoms. *J. Chem. Inf. Model.* **2022**, *62* (11), 2835–2845. https://doi.org/10.1021/acs.jcim.1c01512.

(51) Shemetulskis, N. E.; Dunbar, J. B.; Dunbar, B. W.; Moreland, D. W.; Humblet, C. Enhancing the Diversity of a Corporate Database Using Chemical Database Clustering and Analysis. *J. Comput. Aided. Mol. Des.* **1995**, *9* (5), 407–416. https://doi.org/10.1007/BF00123998.

(52) Böcker, A.; Schneider, G.; Teckentrup, A. NIPALSTREE: A New Hierarchical Clustering Approach for Large Compound Libraries and Its Application to Virtual Screening. *J. Chem. Inf. Model.* **2006**, *46* (6), 2220–2229. https://doi.org/10.1021/ci050541d.

(53) Oliveira, J. C. A.; Frey, J.; Zhang, S.-Q.; Xu, L.-C.; Li, X.; Li, S.-W.; Hong, X.; Ackermann, L. When Machine Learning Meets Molecular Synthesis. *Trends Chem.* **2022**, *4* (10), 863–885. https://doi.org/10.1016/j.trechm.2022.07.005.

(54) Aal E Ali, R. S.; Meng, J.; Khan, M. E. I.; Jiang, X. Machine Learning Advancements in Organic Synthesis: A Focused Exploration of Artificial Intelligence Applications in Chemistry. *Artif. Intell. Chem.* **2024**, *2* (1), 100049. https://doi.org/10.1016/j.aichem.2024.100049.

(55) Shao, J.; Tanner, S. W.; Thompson, N.; Cheatham, T. E. Clustering Molecular Dynamics Trajectories: 1. Characterizing the Performance of Different Clustering Algorithms. *J. Chem. Theory Comput.* **2007**, *3* (6), 2312–2334. https://doi.org/10.1021/ct700119m.

(56) Salo-Ahen, O. M. H.; Alanko, I.; Bhadane, R.; Bonvin, A. M. J. J.; Honorato, R. V.; Hossain, S.; Juffer, A. H.; Kabedev, A.; Lahtela-Kakkonen, M.; Larsen, A. S.; Lescrinier, E.; Marimuthu, P.; Mirza, M. U.; Mustafa, G.; Nunes-Alves, A.; Pantsar, T.; Saadabadi, A.; Singaravelu, K.; Vanmeert, M. Molecular Dynamics Simulations in Drug Discovery and Pharmaceutical Development. *Processes* **2020**, *9* (1), 71.

https://doi.org/10.3390/pr9010071.

(57) Beck, A. G.; Muhoberac, M.; Randolph, C. E.; Beveridge, C. H.; Wijewardhane, P. R.; Kenttämaa, H. I.; Chopra, G. Recent Developments in Machine Learning for Mass Spectrometry. *ACS Meas. Sci. Au* **2024**. https://doi.org/10.1021/acsmeasuresciau.3c00060.

(58) Verbeeck, N.; Caprioli, R. M.; Van de Plas, R. Unsupervised Machine Learning for Exploratory Data Analysis in Imaging Mass Spectrometry. *Mass Spectrom. Rev.* **2020**, *39* (3), 245–291. https://doi.org/10.1002/mas.21602.

(59) McInnes, L.; Healy, J. Accelerated Hierarchical Density Based Clustering. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*; IEEE, 2017; pp 33–42. https://doi.org/10.1109/ICDMW.2017.12.

(60) Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*; Simoudis, E., Han, J., Fayyad, U., Eds.; AAAI Press, 1996; pp 226–231. https://doi.org/10.5555/3001460.3001507.

(61) Williams, F. Point Cloud Utils. 2022. https://www.github.com/fwilliams/point-cloud-utils.

(62) Wang, T. Poisson-Disk Sampling: Theory and Applications. In *Encyclopedia of Computer Graphics and Games*; Springer International Publishing: Cham, 2021; pp 1–8. https://doi.org/10.1007/978-3-319-08234-9_398-1.

(63) 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, É. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

(64) 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.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; Vijaykumar, A.; Bardelli, A. Pietro; Rothberg, A.; Hilboll, A.; Kloeckner, A.; Scopatz, A.; Lee, A.; Rokem, A.; Woods, C. N.; Fulton, C.; Masson, C.; Häggström, C.; Fitzgerald, C.; Nicholson, D. A.; Hagen, D. R.; Pasechnik, D. V.; Olivetti, E.; Martin, E.; Wieser, E.; Silva, F.; Lenders, F.; Wilhelm, F.; Young, G.; Price, G. A.; Ingold, G.-L.; Allen, G. E.; Lee, G. R.; Audren, H.; Probst, I.; Dietrich, J. P.; Silterra, J.; Webber, J. T.; Slavič, J.; Nothman, J.; Buchner, J.; Kulick, J.; Schönberger, J. L.; de Miranda Cardoso, J. V.; Reimer, J.; Harrington, J.; Rodríguez, J. L. C.; Nunez-Iglesias, J.; Kuczynski, J.; Tritz, K.; Thoma, M.; Newville, M.; Kümmerer, M.; Bolingbroke, M.; Tartre, M.; Pak, M.; Smith, N. J.; Nowaczyk, N.; Shebanov, N.; Pavlyk, O.; Brodtkorb, P. A.; Lee, P.; McGibbon, R. T.; Feldbauer, R.; Lewis, S.; Tygier, S.; Sievert, S.; Vigna, S.; Peterson, S.; More, S.; Pudlik, T.; Oshima, T.; Pingel, T. J.; Robitaille, T. P.; Spura, T.; Jones, T. R.; Cera, T.; Leslie, T.; Zito, T.; Krauss, T.; Upadhyay, U.; Halchenko, Y. O.; Vázquez-Baeza, Y. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* **2020**, *17* (3), 261–272.

https://doi.org/10.1038/s41592-019-0686-2.

(65) Bellock, K.; Godber, N.; Khan, P. Alphashape. 2021. https://doi.org/10.5281/zenodo.4697576.

(66) The pandas development team. Pandas. Zenodo 2024. https://doi.org/10.5281/zenodo.10957263.

(67) Gazoni, E.; Clark, C. Openpyxl. 2023. https://foss.heptapod.net/openpyxl/openpyxl.

(68) Landrum, G. RDKit: A Software Suite for Cheminformatics, Computational Chemistry, and Predictive Modeling. 2013.

(69) Moriwaki, H.; Tian, Y.-S.; Kawashita, N.; Takagi, T. Mordred: A Molecular Descriptor Calculator. *J. Cheminform.* **2018**, *10* (1), 4. https://doi.org/10.1186/s13321-018-0258-y.

(70) McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. **2018**. https://doi.org/https://doi.org/10.48550/arXiv.1802.03426.

(71) Harris, C. R.; Millman, K. J.; van der Walt, S. J.; Gommers, R.; Virtanen, P.; Cournapeau, D.; Wieser, E.; Taylor, J.; Berg, S.; Smith, N. J.; Kern, R.; Picus, M.; Hoyer, S.; van Kerkwijk, M. H.; Brett, M.; Haldane, A.; del Río, J. F.; Wiebe, M.; Peterson, P.; Gérard-Marchant, P.; Sheppard, K.; Reddy, T.; Weckesser, W.; Abbasi, H.; Gohlke, C.; Oliphant, T. E. Array Programming with NumPy. *Nature* **2020**, *585* (7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2.

(72) Irwin, J. J.; Sterling, T.; Mysinger, M. M.; Bolstad, E. S.; Coleman, R. G. ZINC: A Free Tool to Discover Chemistry for Biology. *J. Chem. Inf. Model.* **2012**, *52* (7), 1757–1768. https://doi.org/10.1021/ci3001277.

(73) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.* **2018**, *4* (2), 268–276. https://doi.org/10.1021/acscentsci.7b00572.

SUPPORTING INFORMATION

Dedenser: a Python command line tool for clustering and downsampling chemical libraries

Armen G. Beck[1], Jonathan Fine[1,#], Yu-hong Lam[2,#], Edward C. Sherer, Erik L. Regalado[1], Pankaj Aggarwal[1,*]

[1] *Analytical Research & Development, MRL, Merck & Co., Inc., Rahway, NJ, 07065, USA*
[2] *Modeling and Informatics, MRL, Merck & Co., Inc., Rahway, NJ, 07065, USA*
[#] *These authors contributed equally to this work*

[*] Corresponding authors: armen.beck@merck.com (A.G.B.); pankaj.aggarwal@merck.com (P.A.)

Command Line Tool

Beyond the formulation of the Dedenser algorithm and its implementation as a Python library, the Dedenser package has been developed such that it can be used as a command line interface tool. The overall process for using Dedenser through a command line is straightforward, with an example of the overall process of generating a chemical point cloud, downsampling, visualizing, and report generation bellow:

python -m dedenser mkcloud -o <output> <path of SMILES input>

python -m dedenser vis -f -o <output> <point cloud path>

python -m dedenser dedense -o <output> -t <target> <point cloud path>

python -m dedenser mksheet -o <output> -c <point cloud path> -d <downsampled index path>

Dedenser Details

Beyond the general algorithm described prior in the methods and theory section, implementation of Dedenser required various conditions, underlying approaches, and user features to be specified. Briefly significant implementation details can be listed as the following:

- The argument for users to specify employment of convex or optimized concave hulls for calculating cluster volumes. This allows for either the use of potentially more accurate concave hulls/alpha shapes or approximation with convex hulls when clusters contain too many molecules for efficient geometry calculations.
- Saving of calculated Mordred descriptors as a CSV file when generating chemical point clouds.
- Use of default parameters for HDBSCAN excluding the option to modify the minimum cluster size. If users desire to change the HDBSCAN parameters, this can be remedied by the simple redefining of desired parameter values in a single line of source code.
- Use of default parameters for UMAP embeddings, as the default parameters are typically satisfactory and skilled users may simply generate and provide their own chemical point clouds to their desired specificities.
- Users provide the proportionality of downsampling desired (some value between 0 & 1). The target number of molecules to keep is calculated from this user defined proportionality value.
- If the calculated number of points of a cluster to keep is less than 15, then random sampling is used to select the members of the cluster that are kept as well as one which is taken as the nearest neighbor of a fitted Gaussian's mean (cluster centroid). This is to aid in downsampling accuracy, as Poisson disk sampling may yield greater or fewer molecules than the target with proportionately significant disparity, due to spatial constraints, when the target value is this low.
- If the calculated number of points of a cluster to keep rounds to 0, then the point closest to the cluster centroid is kept.

Table S1. Command line interface arguments and commands for Dedenser.

Flag/Command	Description
-h [--help]	Prints details regarding options/arguments.
-o [--path_out]	Path/file for output results Should not have file extensions i.e. '.xlxs'.
-f [--fig]	If used, saves figures rather than just displaying.
-d [--down]	Path/file for downsample list. Used with secondary commands 'vis' and 'mksheet'.
-a [--alpha]	Use optimized alpha shapes for cluster volume estimation.
-s [--sep]	Specify separator for SMILES file.
-p [--pos]	Specify position for SMILES in SMILES file. Note: position is zero indexed.
-r [--rand]	Random seed for downsampling.
-m [--min]	The 'min_size' parameter for HDBSCAN.
-t [--target]	Target downsampling percentage.
-dw [--dweight]	Weighting term for density bias. If used, density based downsampling is utilized over volume.
-vw [--vweight]	Weighting term for volume bias. If used, volume based downsampling is utilized with weighting of clusters.
-c [--cloud]	Path/file for chemical point cloud (only when using 'mksheet' command).
-x [--excel]	Use to load and save excel sheets rather than delimited text. Default is False.
-H [--header]	Specify if a header is present when loading sheets/delimited text. Default is False
-S [--strict]	Completely drops clusters with 'target values' of 0 rather than keeping a single molecule.
--SHOW	Display HDBSCAN clustering results prior to downsampling.
mkcloud	Generate a chemical point cloud with Mordred & UMAP given a file with SMILES.
dedense	The main downsampling algorithm in Dedenser.
vis	Visualize chemical point clouds either prior or after downsampling.
mksheet	Generate a report sheet containing SMILES and respective UMAP (3D) coordinates.

```
Input  : ChemicalPointcloud, targetPercentage
Output: selectedPoints
(1)  targetMembers ← Size(ChemicalPointcloud) * targetPercentage;
(2)  noise, clusters ← HdbscanClustering(ChemicalPointcloud);
(3)  clusterVolumes ← Hull(clusters);
(4)  remainingPoints ← targetMembers - noise;
(5)  selectedPoints ← noise;
(6)  if noise < targetMembers * 0.95 then
(7)      noiseVolume ← Hull(noise);
(8)      noiseTarget ← noiseVolume/Sum(clusterVolumes,noiseVolume);
(9)      selectedPoints ← PoissonDiskDownsampling(noise,
          noiseTarget);
(10)     remainingPoints ← targetMembers - selectedPoints;
(11) CheckClusters ← True;
(12) while CheckClusters is True do
(13)     CheckClusters ← False;
(14)     for cluster in clusters do
(15)         targetMembership ← remainingPoints *
              clusterVolumes[cluster]/Sum(clusterVolumes);
(16)         if targetMembership > len(cluster) then
(17)             selectedPoints += cluster;
(18)             Pop(clusters,cluster);
(19)             CheckClusters ← True
(20)     remainingPoints ← targetMembers - selectedPoints;
(21) for cluster in clusters do
(22)     targetMembership ← remainingPoints *
          clusterVolumes[cluster]/Sum(clusterVolumes);
(23)     if targetMembership > 0 then
(24)         if targetMembership < 15 then
(25)             selectedPoints += GetCentroidNeighbor(cluster);
(26)             selectedPoints += RandomSelection(cluster,
                  targetMemberships - 1);
(27)         else
(28)             selectedPoints += PoissonDiskDownsampling(cluster,
                  targetMemberships[cluster]);
(29)     else
(30)         selectedPoints += GetCentroidNeighbor(cluster);
(31) return selectedPoints;
```

Algorithm S1. Pseudocode for volume based downsampling with the Dedenser algorithm. Given a chemical point cloud and a target percentage to be downsampled to, Dedenser clusters the point cloud with HDBSCAN and subsequently selects cluster members via Poisson disk downsampling. Note that the calculation of target cluster memberships (lines 15 & 22) display the volume based approach described in eq. 2, and the weighted density based approach described in eq. 3 & 4 are implemented options in the Dedenser software package.

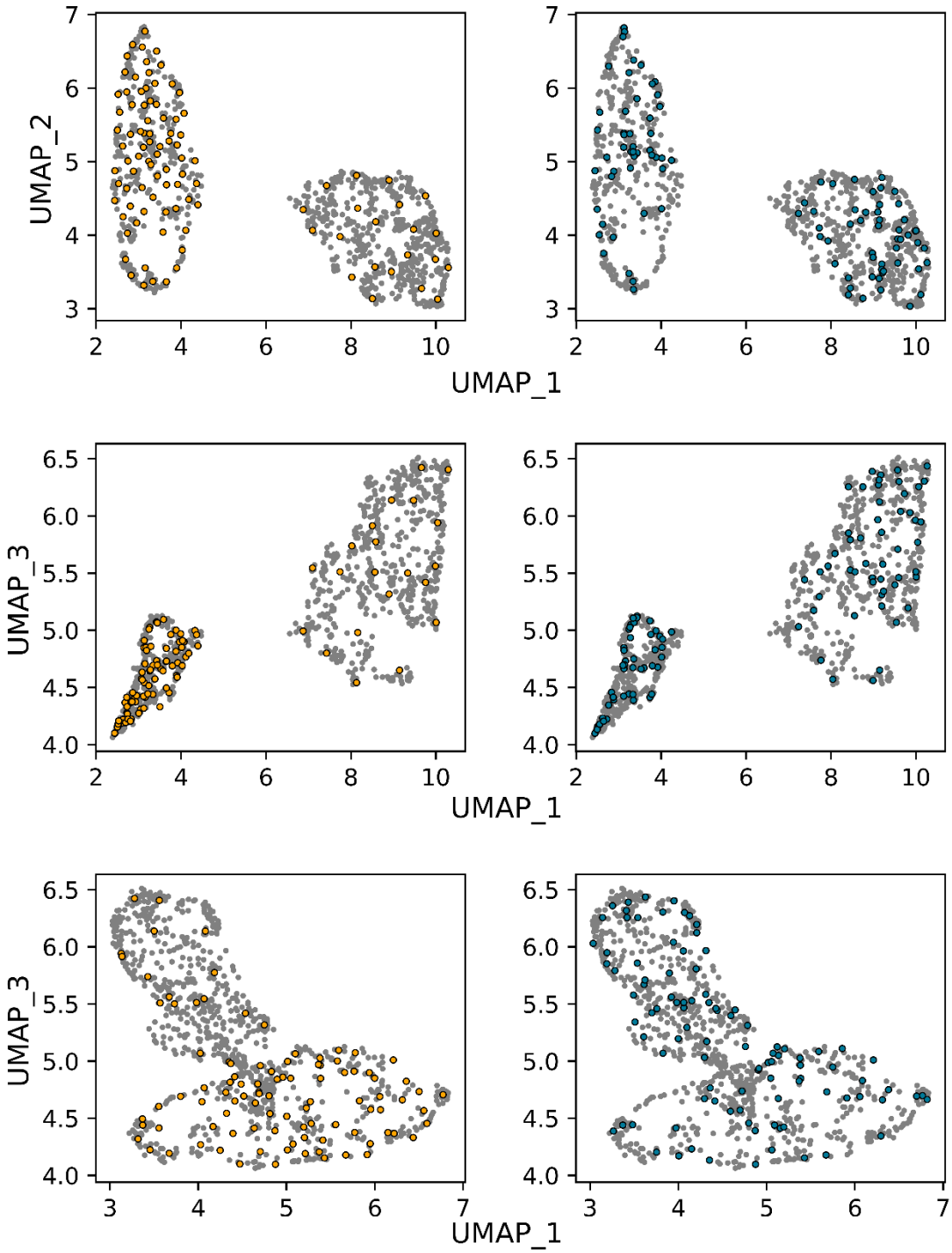

Figure S1. UMAP plots of 1000 ZINC compounds (grey) where orange scatter points in the left column displaying 99 downsampled compounds using Dedenser with a density weighting of 5 while blue scatter points in the right column display 99 randomly selected compounds.

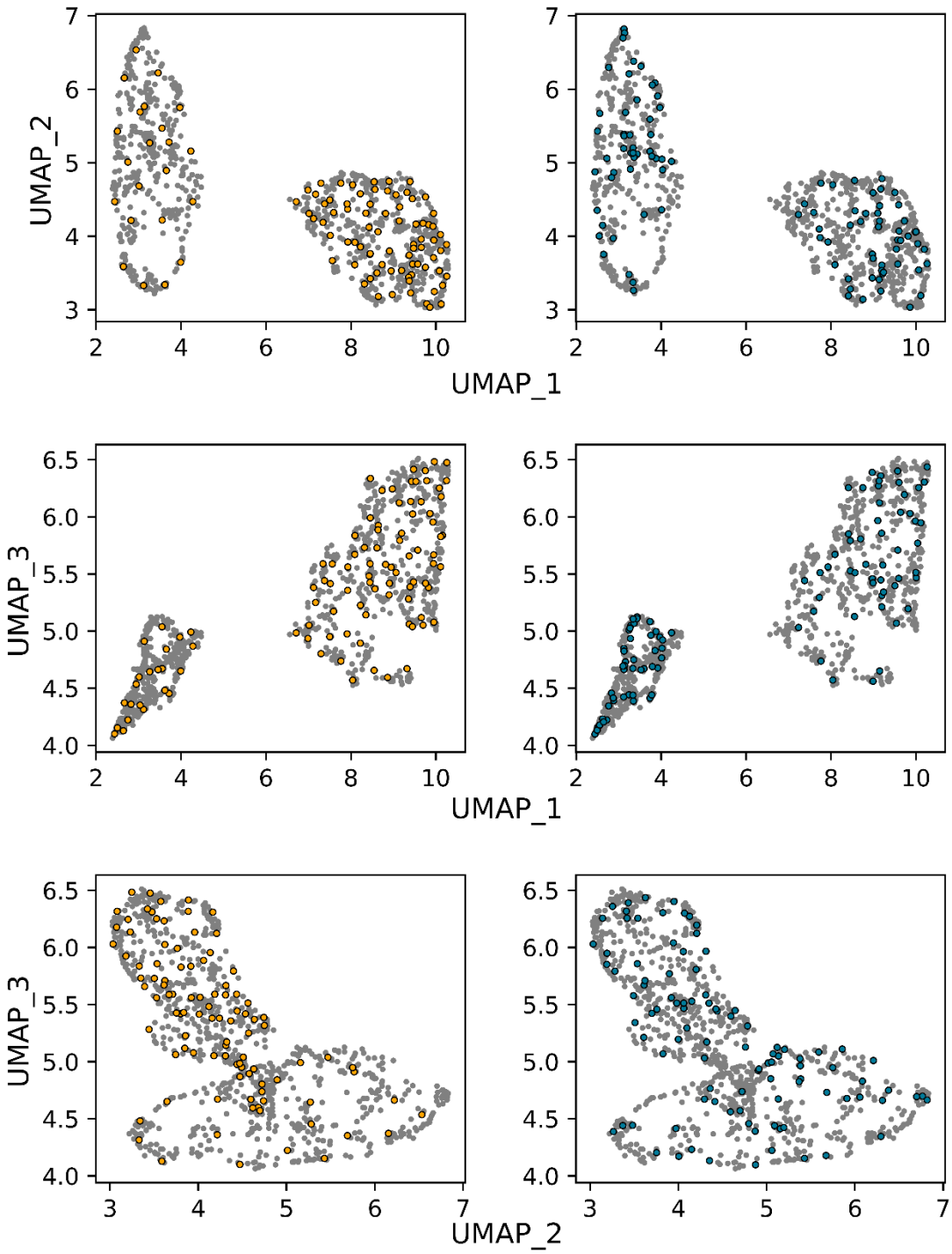

Figure S2. UMAP plots of 1000 ZINC compounds (grey) where orange scatter points in the left column displaying 99 downsampled compounds using Dedenser with a density weighting of -5 while blue scatter points in the right column display 99 randomly selected compounds.

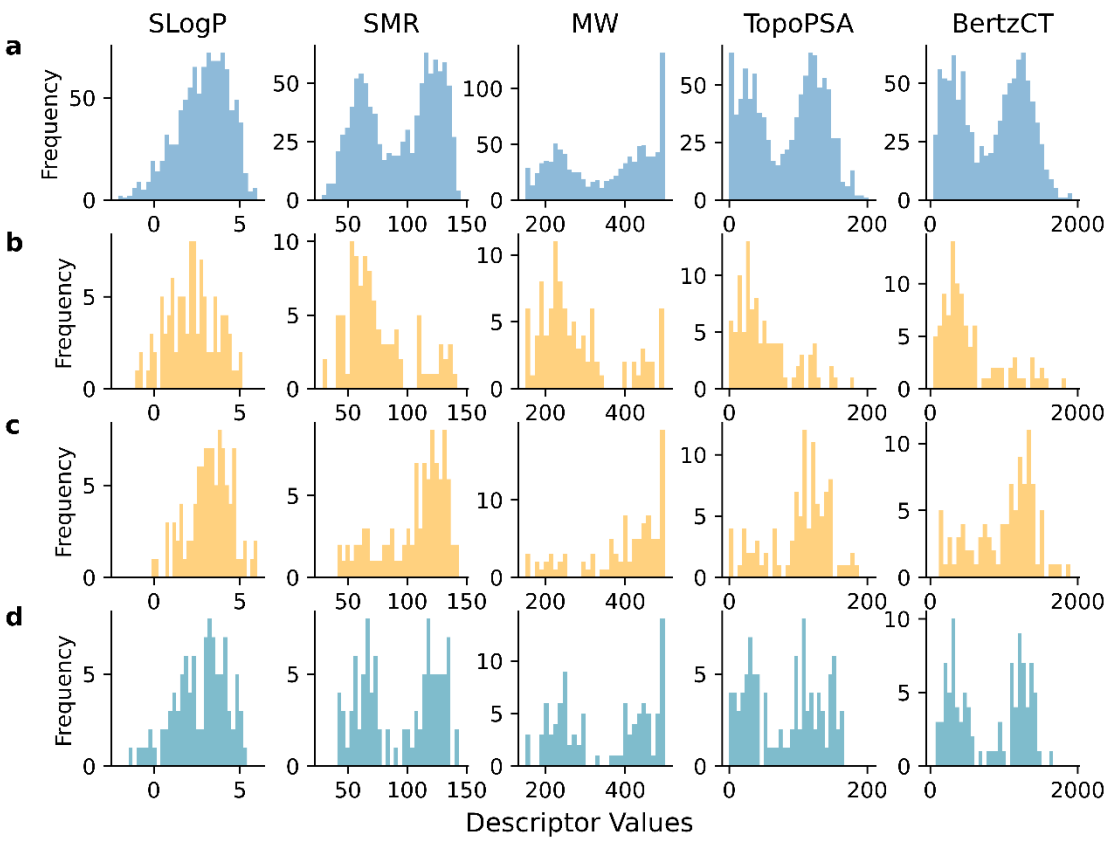

Figure S4. Molecular descriptors for a 1000 molecules from ZINC and downsampled subsets displaying distributions of Wildman-Crippen LogP (SLogP), Wildman-Crippen molar refractivity (SMR), molecular weight (MW), topological polarizable surface area (TopoPSA), and Bertz index (BertzCT). (a) Top in blue are the distributions of the initial 1000 molecules. In the middle in orange are the distributions of 99 molecules downsampled using Dedenser with density weightings of (b) 5 and (c) -5. (d) Bottom in blue are the distributions of 99 molecules selected randomly.

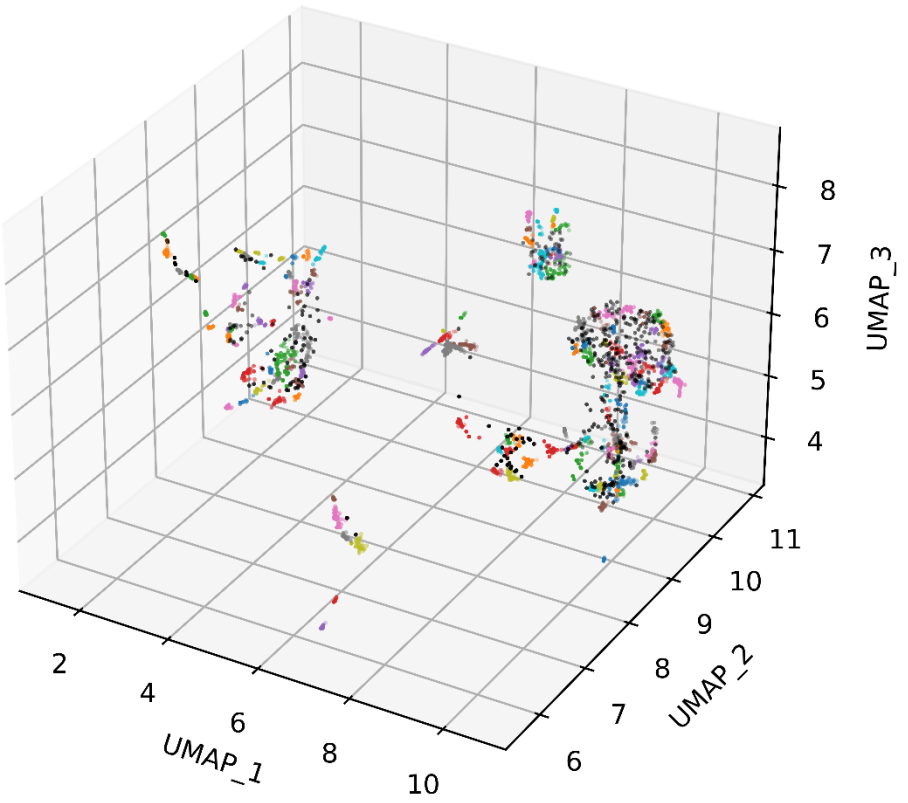

Figure S5. Clustering of aryl bromides using default Dedenser parameters and the visualization flag prior to downsampling. Note that cluster colors are arbitrary and are not unique due to the vast number of possible clusters. Black scatter points represent chemical singletons considered as noise by HDBSCAN.

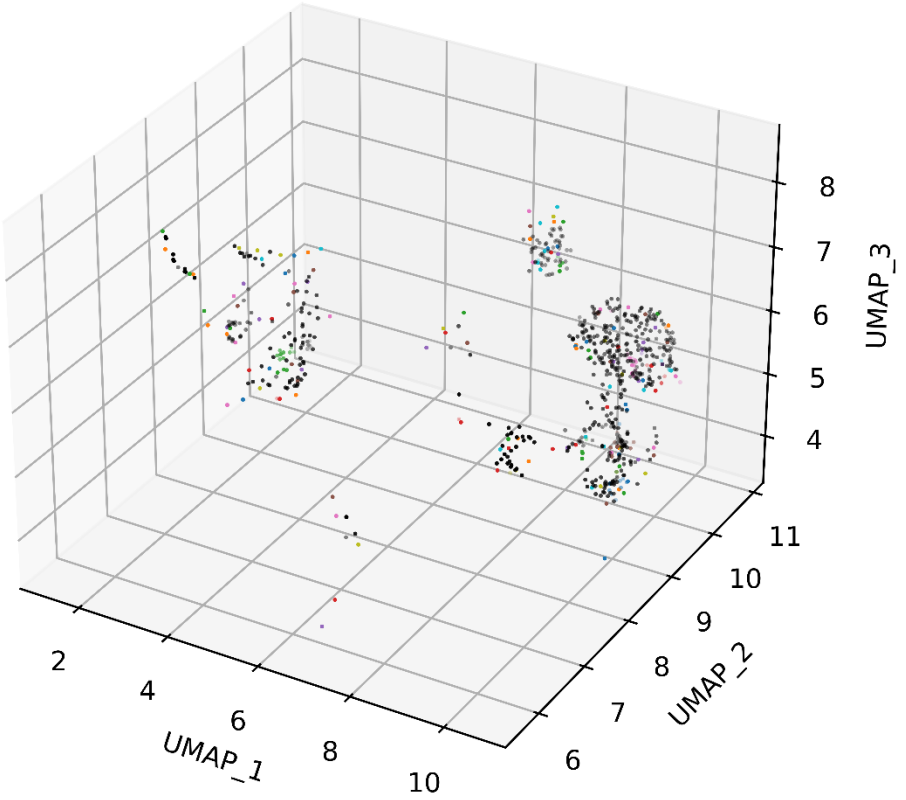

Figure S6. Clustering of aryl bromides using default Dedenser parameters and the visualization flag post downsampling. Note that cluster colors are arbitrary and are not unique due to the vast number of possible clusters. Black scatter points represent chemical singletons considered as noise by HDBSCAN.

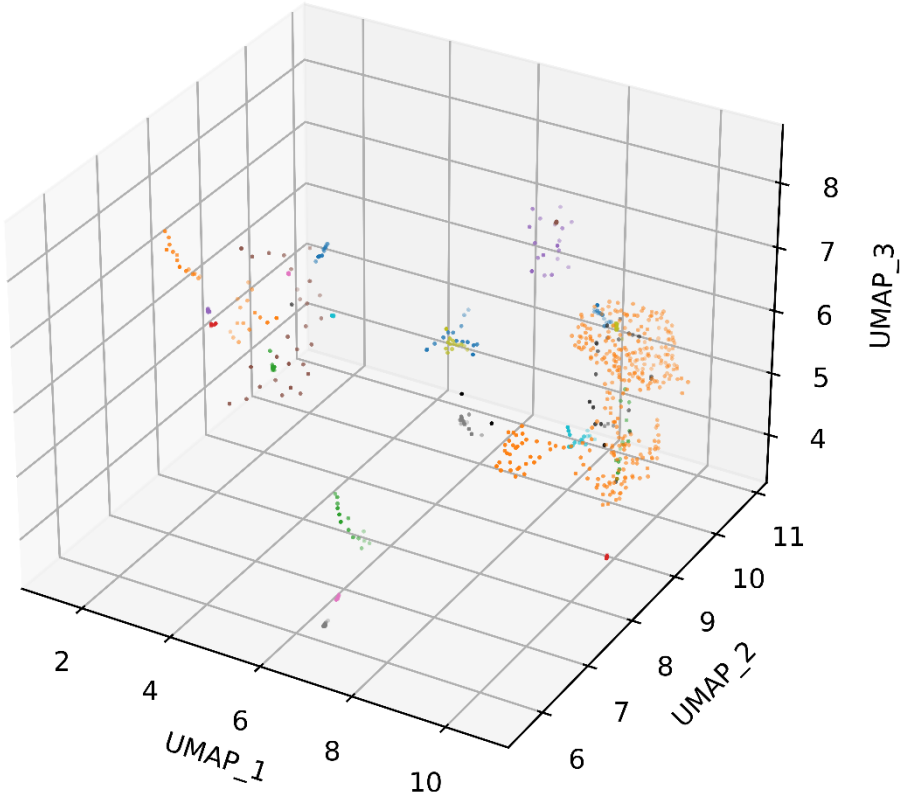

Figure S7. Clustering of aryl bromides using Dedenser with epsilon set to 0.22, volume weighting set to 4, and the visualization flag post downsampling. Note that cluster colors are arbitrary and are not unique due to the vast number of possible clusters. Black scatter points represent chemical singletons considered as noise by HDBSCAN.