

ChemPlot, a Python library for chemical space visualization

Murat Cihan Sorkun^{1,2,3}, Dajt Mullaj^{1,2}, J. M. Vianney A. Koelman^{1,2,3}, and Süleyman Er^{1,2*}

Visualizing chemical spaces streamlines the analysis of molecular datasets by reducing the information to human perception level, hence it forms an integral piece of molecular engineering, including chemical library design, high-throughput screening, diversity analysis, and outlier detection. We present here ChemPlot, which enables users to visualize the chemical space of molecular datasets in both static and interactive ways. ChemPlot features structural and tailored similarity methods, together with three different dimensionality reduction methods: PCA, t-SNE, and UMAP. ChemPlot is the first visualization software that tackles the activity/property cliff problem by incorporating tailored similarity. With tailored similarity, the chemical space is constructed in a supervised manner considering target properties. Additionally, we propose a metric, the Distance Property Relationship score, to quantify the property difference of similar (i.e. close) molecules in the visualized chemical space. ChemPlot can be installed via Conda or PyPI (pip) and a web application is freely accessible at <https://www.amdlab.nl/chemplot/>.

INTRODUCTION

Data visualization is an essential tool for scientists to reveal patterns that are hidden in high-dimensional data, to interpret these patterns, and - as a matter of usual practice - to convey them to non-experts. Moreover, in the current era of artificial intelligence (AI)-guided chemical design, visualization is frequently employed to describe the applicability domains of data-driven models. In molecular data science, the complex variability of molecular data, as stored in high-dimensional spaces spanned by graphs, textual representation, atom coordinates, or any combinations thereof, demands the development of chemically informative data visualization methods and tools. The data visualization provides a practical means to reduce the high-dimensional molecular data to spaces of two (2D) or three dimensions (3D). A visual inspection of data in low-dimensional chemical space enables a more realistic screening of molecules with user-desired properties, in addition to exploratory analysis of the data including, but not limited to, diversity and outlier detection.

Chemical space of molecules is a multi-dimensional space in which the alike molecules are grouped together based on their similarities. The chemical space terminology is commonly used in two distinct, yet related, concepts. First, it denotes the property space of an unknown size, which is spanned by all molecules that exist or are likely to exist. Secondly, it also finds use in the context of a given molecular dataset with a finite size. In the current study, we refer to the latter terminology of chemical space and focus on quantifying the similarities of molecules within domain-specific datasets.

¹DIFFER – Dutch Institute for Fundamental Energy Research, De Zaale 20, Eindhoven 5612 AJ, the Netherlands.

²CCER - Center for Computational Energy Research, De Zaale 20, Eindhoven 5612 AJ, the Netherlands.

³Department of Applied Physics, Eindhoven University of Technology, Eindhoven 5600 MB, the Netherlands. *email: s.er@differ.nl

Visualization of chemical spaces usually consists of three main steps. In the first step, molecular representations (e.g. SMILES or InChI) are converted into binary or real-value arrays. Each element of the array corresponds to a dimension in the chemical space. In the second step, the high-dimensional chemical space is reduced to 2D or 3D spaces. In the final step, the reduced chemical space is converted into a visual representation. Fingerprints and descriptors¹⁻³ are the most widely used molecular encoding methods for the first step. Principal component analysis (PCA)⁴, self-organized maps (SOM)⁵, multidimensional scaling (MDS)⁶, and t-distributed stochastic neighbor embedding (t-SNE)⁷ are the commonly used dimensionality reduction methods. Finally, visualizations can take the shape of either 2D or 3D, static or interactive, scatter or grid-like plots. In addition to these, there are alternative chemical space visualization approaches including the minimum spanning-trees⁸, similarity networks⁹, and uniform manifold approximation and projection (UMAP)¹⁰.

In practice, screening of molecules with desired properties is usually based on the similarity property principle (SPP), which assumes that similar molecules (close to each other in chemical space) show similar properties. However, it is not uncommon for pairs of similar molecules to violate the SPP principle¹¹ and such violations are referred to as activity/property cliffs (APCs). APCs may mislead the screening and result in a deceleration of the discovery process of molecules with target properties. It is important to spot or, even better, prevent APCs to facilitate a speedy discovery process. Several visualization tools include additional analysis modules for APCs¹²⁻¹⁴. However, currently, there is no publicly available chemical space visualization tool that provides a remedy to the APC problem as part of the process.

Even though the majority of the chemical space visualization tools are distributed as an integrated feature of commercial software packages, several dedicated tools are also freely available¹⁵. In support of open science, the latter tools have been extensively serving the chemical informatics community, although admitting that they carry the non-negligible sustainability concerns. For instance, free visualization tools are available as standalone software^{12, 13, 16, 17} and web server applications^{14, 18, 19}. However, they all require developer-dependent maintenance for version updates, bug-fixing, and adding new features. Therefore, with the completion of code development projects, the once-effective tools may become inaccessible, outmoded, or face compliance issues with the continually evolving operating systems.

Here, we present ChemPlot, a Python library for chemical space visualization. ChemPlot is free and its source code is available on GitHub under the Berkeley Source Distribution (BSD) license. ChemPlot is a user-friendly software, in which users can transform their datasets into interactive 2D chemical space visualizations. The current version (1.2.0) of ChemPlot provides three dimensionality reduction methods: PCA, t-SNE, and UMAP. Additionally, two similarity methods are supported: structural and tailored. According to our knowledge, ChemPlot is the first tool that applies tailored similarity²⁰, which measures the similarity of molecules in a supervised manner considering the target properties of compounds, and thus provide a way to deal with the APC problem. Additionally, to better quantify the target property difference between molecules that are close to each other in a chemical space, we devise a new metric called the Distance Property Relationship (DPR) scores. ChemPlot can be installed via well-known Python package managers, including Conda and PyPI (pip). Finally, alongside the user manual that contains practical information and examples, we also provide a web interface to encourage the use of ChemPlot without the

need for coding expertise (see Code Availability).

RESULTS

In this section, we present the features of ChemPlot with help of example codes and figures. We start with the installation of ChemPlot, next exemplify various data visualizations, and then present the advanced functionalities of the code. Additionally, we compare the visualization performance of the currently available dimension reduction and similarity methods in ChemPlot. Lastly, we illustrate a workable use of the ChemPlot web application.

Installation

ChemPlot is an open-source Python library with its source code stored on a GitHub repository (see Code Availability). In addition to a direct installation from the repository, users can also install the deployed packages via Conda and PyPI package management environments. ChemPlot runs on the Python framework and it can be installed from two alternative sources via the command prompt using either of the following commands.

```
conda install -c conda-forge chemplot
pip install chemplot
```

Visualization

To visualize the chemical space of a dataset, users first need to construct a *Plotter* object. The *Plotter* object can be created from a molecular dataset that contains SMILES or InChI notations of compounds. The below code shows a basic example of t-SNE plotting from the SMILES data.

```
from chemplot import Plotter

cp = Plotter.from_smiles(smiles_list)
cp.tsne()
cp.visualize_plot()
```

Figure 1a and 1c show the representative visualizations of the chemical space of β -secretase 1 (BACE)²¹ and Lipophilicity²² datasets as generated by using the t-SNE method with default parameters.

Target Coloring

ChemPlot allows coloring the molecules based on a given target property. The given property is automatically classified as a numerical or categorical variable and the coloring is applied accordingly. Alternatively, users can also specify the target type themselves. As an example, the following code shows how the target assignment can be applied while creating the *Plotter* object.

```
cp = Plotter.from_smiles(smiles_list, target=target_list, target_type="C")
```

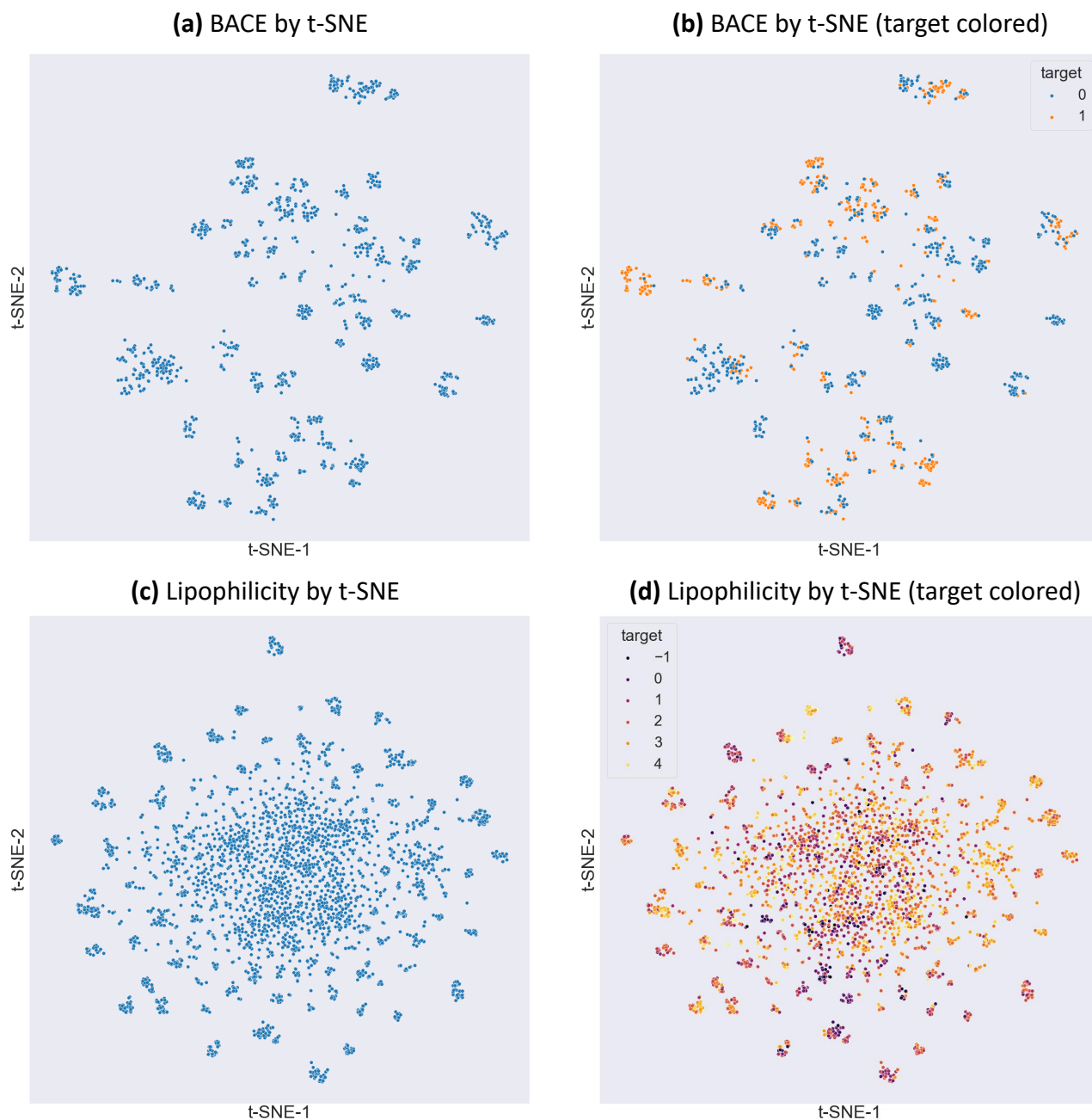


Figure 1. BACE (**a** and **b**) and Lipophilicity (**c** and **d**) datasets visualized by using ChemPlot. In **a** and **c**, all the data points are shown with a single color, whereas in **b** and **d**, they are colored according to the targets' values. In **b**, the target properties are categorical values and therefore the points are colored according to the category of the target. In **d**, the target properties are continuous values and therefore the points are colored according to the corresponding color from the predefined colormap. The corresponding values for the class colors and the gradient intervals are shown in the legends.

Figure 1b and 1d show the representative visualizations of the chemical space of BACE (categorical) and Lipophilicity (numerical) datasets as generated using the t-SNE method and colored with reference to the targets.

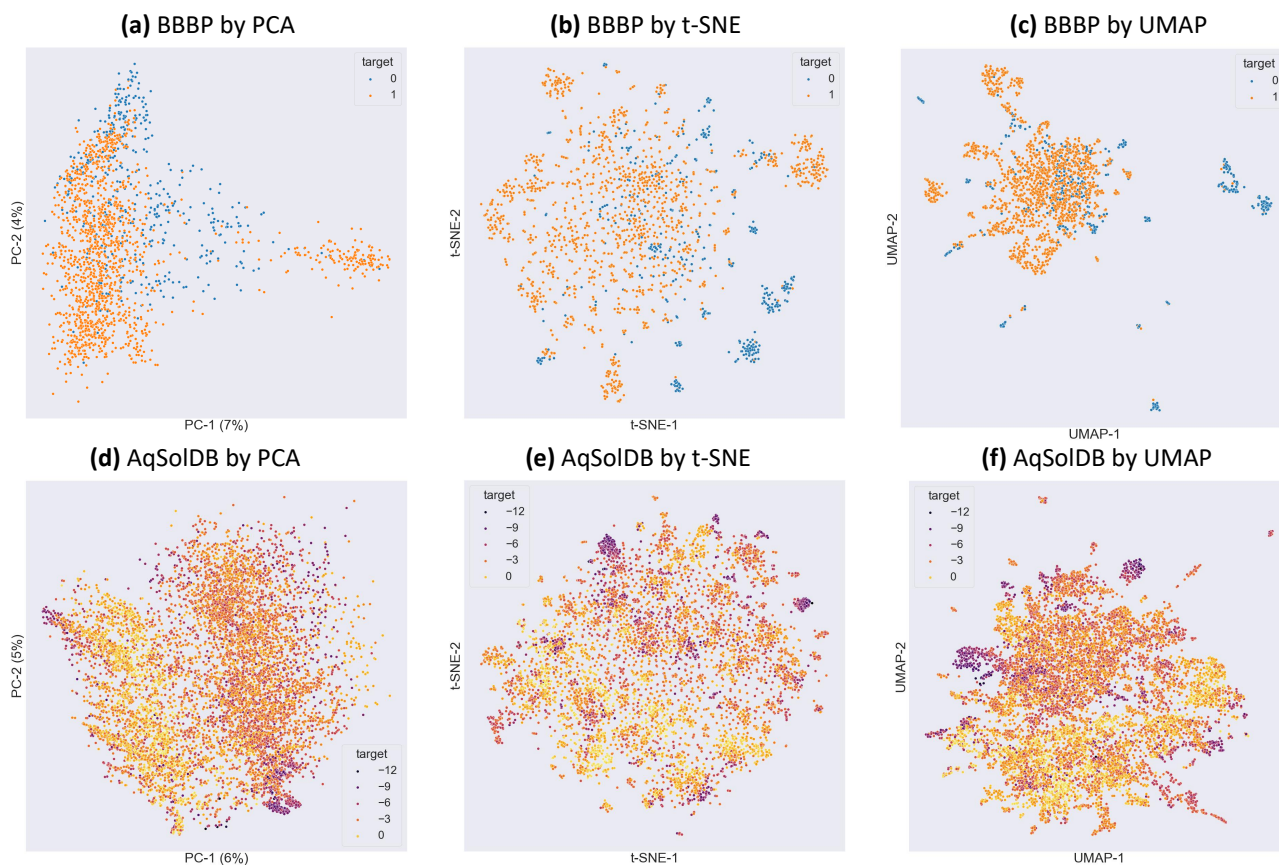


Figure 2. BBBP (a, b, and c) and AqSolDB (d, e, and f) datasets visualized by using structural similarity. In a and d, the dimensions are reduced by PCA. The first and the second principal components are used as the horizontal and vertical axes, respectively. The explained variances of each component are given in parenthesis. In b and e, the dimensions are reduced by t-SNE. In c and f, the dimensions are reduced by UMAP.

Dimensionality Reduction

In ChemPlot, three different dimensionality reduction methods (PCA, t-SNE, and UMAP) can be applied to map the molecules onto 2D. Figure 2 shows the representative visualizations of the Blood-brain barrier penetration (BBBP)²³ and AqSolDB²⁴ datasets that have been generated by applying these three methods. While PCA provides a linear projection of given dimensions, both t-SNE and UMAP apply non-linear 2D mappings by clustering and locating molecules depending on their local neighborhoods. For the t-SNE and UMAP parameters, users can choose between the two available options of: *i*) resorting to the optimized values that are calculated and assigned automatically by ChemPlot, or *ii*) assigning the values themselves. Additional technicalities of the dimensionality reduction algorithms are provided in the Methods section. The following code shows the execution of the three different dimensionality reduction methods that are currently available in ChemPlot.

```
cp.pca() // cp.tsne() // cp.umap()
```

Molecular Similarity

ChemPlot implements two types, structural and tailored, of molecular similarity methods. Structural similarity uses the

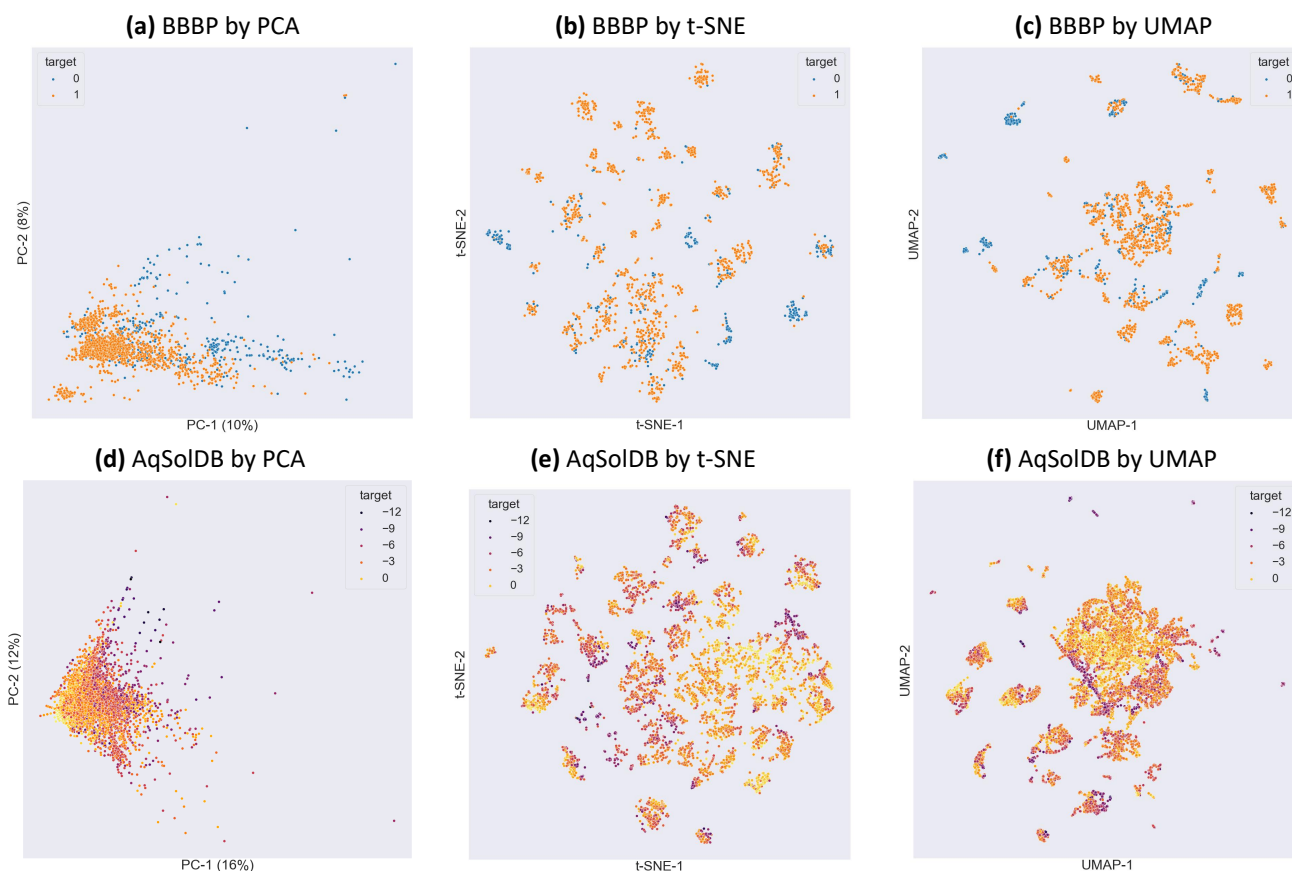


Figure 3. BBBP (a, b, and c) and AqSolDB (d, e, and f) datasets are visualized in the same layout as in Figure 2, but by using tailored similarity. Compared to the structural similarity, for the plots reduced by PCA, the explained variances are higher but at the same time the usage of the space is less efficient. For the plots reduced by t-SNE and UMAP, the clusters are separated from each other more clearly.

generated molecule substructures and ignores the target property when evaluating the similarity. Tailored similarity, on the other hand, uses only the descriptors that correlate with the target property. Therefore, while structural similarity produces more generic multi-purpose chemical spaces, tailored similarity produces property-sensitive focused chemical spaces. Figure 2 and Figure 3 show visualizations of the same datasets by employing structural and tailored similarity methods, respectively. In ChemPlot, the similarity type can be assigned while creating the *Plotter* object. The default value for the similarity type is "tailored" when a target list is provided, otherwise, it is "structural". The following code shows the similarity type assignment.

```
cp = Plotter.from_smiles(smiles_list, target=target_list, sim_type="structural")
```

Clustering

ChemPlot features clustering of molecules in the reduced chemical space. The total number of clusters is defined by the *n_cluster* parameter. In the visuals, a legend is included which shows the assigned color, cluster number, and share of coverage over the whole dataset for each cluster. Alternatively, users can interactively select and recolor clusters of their choice to visually distinguish them. This option is particularly convenient for evaluating the extrapolation performance of ML models,

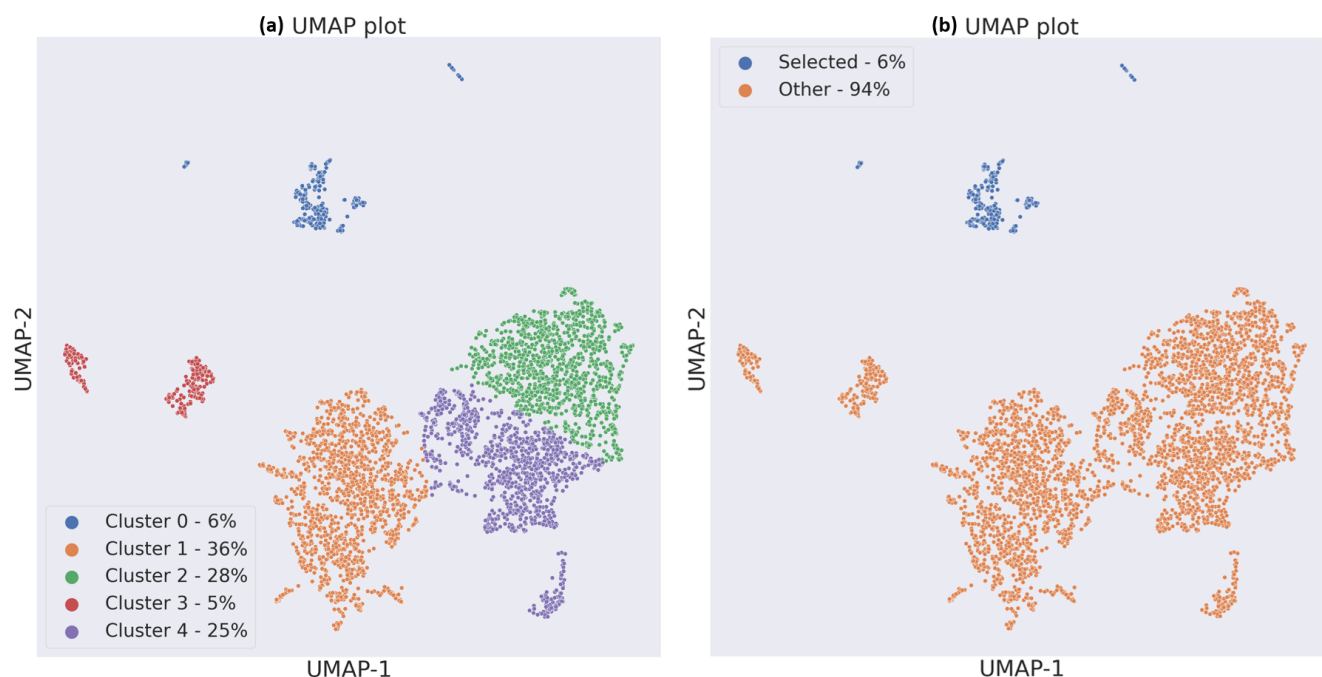


Figure 4. The clustering of sub-spaces from the Lipophilicity dataset. In (a), the chemical space is distributed over five clusters. The colors, identifiers, and allotments of the clusters are included in the legend. In (b), the user-selected cluster, with blue color and id=0, is clearly distinguishable from the remaining clusters, shown with orange color, of the dataset.

such as by setting aside a test set that is positioned away from the training set. In addition to this, highlighting specific clusters or singletons could also provide a useful way when exploring and learning on the chemical space of the dataset. Figure 4 shows (a) clustered chemical space, and (b) user-highlighted clusters for the Lipophilicity²² dataset. The following code illustrates how to use ChemPlot for clustering the chemical space and selecting specific clusters.

```
cp = Plotter.from_smiles(smiles_list)
cp.umap()
cp.cluster(n_cluster=5)
cp.visualize_plot(clusters=True) #visualize all clusters
cp.visualize_plot(clusters=0)   #highlight selected cluster
```

Plotting Options

ChemPlot supports both static and interactive plotting. The static plots are images that can be exported in various file formats, including PNG, JPEG, PDF, and SVG. The visualizations shown in Figures 1-4 are examples of static plots. Differently, interactive plots allow users to drag, zoom, and highlight sections of the visualized data. Hovering over the data points displays a tool-tip that contains the 2D chemical drawing and the target value of the molecule for that data point. Users have the option of saving the images of their plots both in their original and newly edited states. Furthermore, the interactive plot is created in HTML format, so that the users can export and embed it, for instance, to their own websites. Figure 5 shows an example interactive plot of the BBBP dataset. The following code shows the interactive plot generation of the chemical space that has

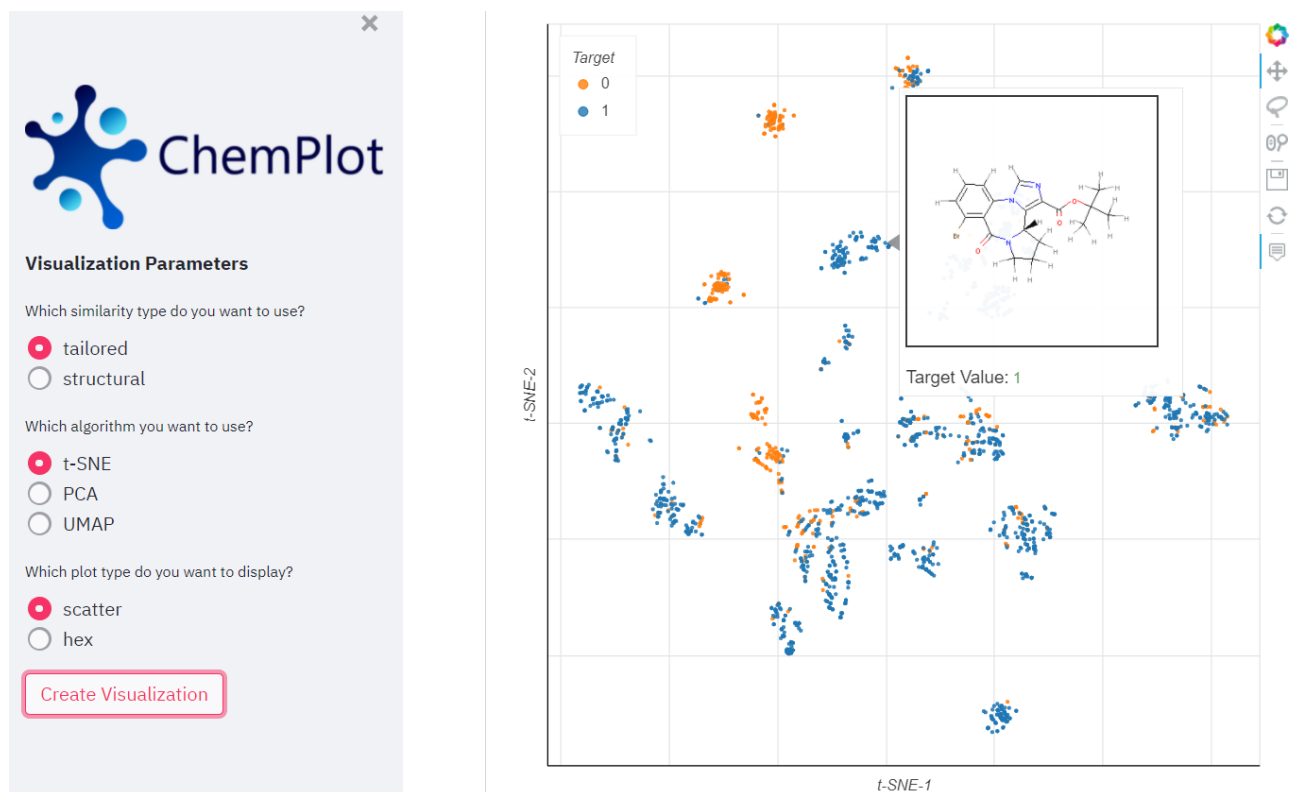


Figure 5. A screenshot of the ChemPlot web application that is used to interactively visualize the BBBP dataset. The left panel contains the configuration options. The right panel contains the interactive plot that is created according to a selected configuration. Hovering over the data points displays a tooltip containing the 2D image of the molecule. From the top-right panel, users can activate the drag, highlight, zoom, save, and reset functions.

dimensionally been reduced by using the PCA method.

```
cp.pca ()
cp.interactive_plot (show_plot=True)
```

Additional Features

ChemPlot contains stochastic processes (t-SNE and UMAP) that generate different visualizations for each run. In order to create reproducible plots, users can set the random state parameter as shown below.

```
cp.umap (random_state=0)
```

Using ChemPlot, the users can identify and remove the outlier molecules. This option is controlled by a boolean parameter, which is by default set to "False". To remove the outliers in data, users can set this parameter to "True" as shown below.

```
cp.visualize_plot (remove_outliers=True)
```

To speed up the plotting process for the large datasets, ChemPlot comes with a PCA pre-reduction option that reduces the number of dimensions to 10 before applying the t-SNE or UMAP dimensionality reduction methods. Users can set the parameter to "True" to include the PCA pre-reduction step.

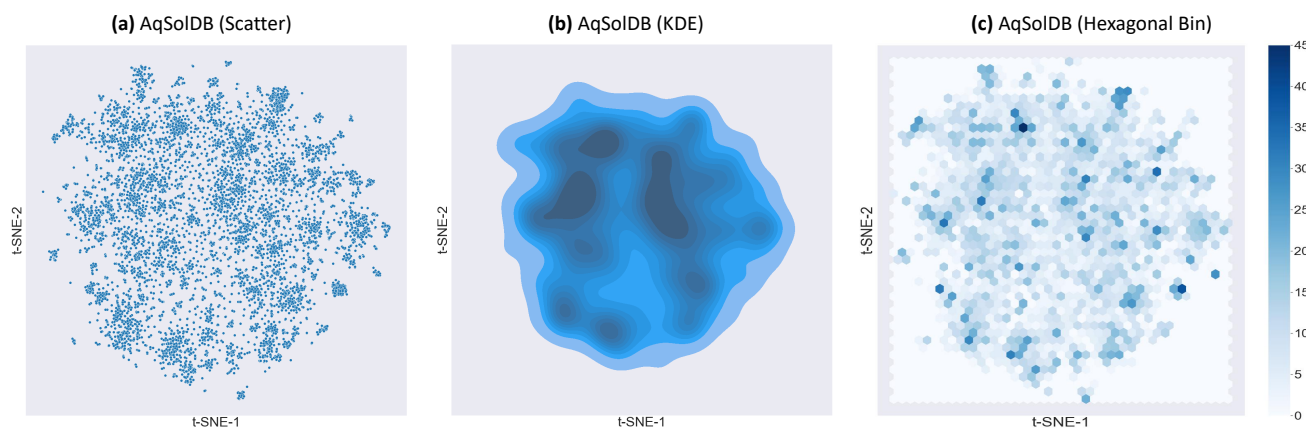


Figure 6. The plotting types found in ChemPlot, as demonstrated by applying the t-SNE and structural similarity methods for the visualization of the AqSolDB dataset. In **a**, the scatter plot is shown, where each point corresponds to a molecule. In **b**, the KDE plot is shown, where light and dark colors show relatively low- and high-density regions of molecules, respectively. In **c**, the hexagonal bin plot is shown, where the color of hexagon indicates the density of molecules it contains. A color scale is provided on the right edge of the figure showing the relation between the representative colors and the number of molecules.

```
cp.umap(PCA=True)
cp.tsne(PCA=True)
```

ChemPlot produces square images with a default size of 20×20 inches and 700×700 pixels for static and interactive plots, respectively. Alternatively, users can define custom sizes for either of their static and interactive plots as shown with the below piece of code.

```
cp.visualize_plot(size=20)
cp.interactive_plot(size=700)
```

In ChemPlot, the default plotting type is Scatter. In addition, ChemPlot provides Kernel Density Estimation (KDE) and Hexagonal plotting options. Figure 5 shows the visualization of the same dataset by the three plotting types. Users can set the plotting type as shown below.

```
cp.visualize_plot(kind="kde")
cp.visualize_plot(kind="hex")
```

Software Testing

Software testing is an indispensable step for robust and sustainable software development. For this purpose, we implemented continuous integration by adding an automated unit testing workflow into the library's repository. Unit tests aim to isolate each individual unit or module of the software and validate whether they perform as expected or not. The test workflow contains a total of 270 test cases that cover all combinations of parameters for all the available methods in ChemPlot. The total line coverage of the unit tests is 99%, as validated by Coveralls²⁷ automated test runs. The tests are executed on three operating systems (latest versions provided by GitHub) including Linux (Ubuntu), Microsoft Windows Server, and macOS for the four

Dataset	Size	Target	Num. of Dim.	Similarity	Construction	PCA	t-SNE	UMAP
FreeSolv ²⁵	642	numerical	30	tailored	2.25	0.02	1.73	2.88
FreeSolv ²⁵	642	-	991	structural	0.85	0.10	1.86	2.90
Huuskonen ²⁶	1291	numerical	16	tailored	6.35	0.03	5.08	4.12
Huuskonen ²⁶	1291	-	1684	structural	1.99	0.41	4.99	4.42
BACE ²¹	1513	numerical	25	tailored	11.37	0.01	4.67	4.29
BACE ²¹	1513	-	1690	structural	2.61	0.45	5.18	4.99
Lipophilicity ²²	4200	numerical	17	tailored	27.31	0.04	16.42	18.09
Lipophilicity ²²	4200	-	2035	structural	6.93	1.66	19.77	7.00
AqSolDB ²⁴	9982	numerical	23	tailored	47.95	0.08	52.23	9.34
AqSolDB ²⁴	9982	-	2046	structural	14.87	3.90	64.27	17.50
ClinTox ²²	1478	categorical	69	tailored	9.82	0.04	4.67	4.23
ClinTox ²²	1478	-	2007	structural	2.41	0.54	6.59	5.51
BACE ²¹	1513	categorical	63	tailored	11.50	0.02	4.76	4.32
BACE ²¹	1513	-	1690	structural	2.66	0.44	5.24	5.04
BBBP ²³	2039	categorical	69	tailored	12.34	0.03	6.84	5.93
BBBP ²³	2039	-	2016	structural	3.26	0.74	8.58	7.37
HIV ²²	41127	categorical	126	tailored	453.23	1.28	294.90	28.06
HIV ²²	41127	-	2048	structural	74.20	19.38	429.01	179.83

Table 1. The performance results of ChemPlot for the various configurations on the benchmark datasets. The elapsed-time (s) is shown separately for the construction and reduction phases.

latest major versions of Python (3.6—3.9). The current version of ChemPlot successfully passes all the unit tests. Moreover, the unit tests are packaged with the library so that the users can verify their installations locally. Furthermore, the automated tests are build and run when a new change is pushed into the main repository.

In addition to the automated test workflow, we also developed exclusive visualization and performance tests. These tests do not check the correctness of the code, but instead, they yield new insights into the overall quality of the software outputs. The visualization and performance tests are not automated. However, by running the provided test files in the GitHub repository, the users can still perform these tests manually in their own environments.

Visualization tests aim to validate whether the visualization output is properly displayed. For this purpose, we developed a test code that creates a PDF file of plots, which are generated by using different parameters according to the given datasets of molecules. The produced PDF file is inspected and validated by the tester. The current version of ChemPlot is tested on three different molecular datasets^{21,23,26} and all the generated plots successfully passed the visual inspection tests.

Performance tests aim to evaluate the speed of the software under different circumstances. For this purpose, we developed a test code to gauge the visualization performance of ChemPlot for the well-known molecular datasets^{21–26} of sizes between 642 and 41,127 molecules. The performance metrics of ChemPlot on an ordinary personal computer (processor with 2.80 GHz clock speed and 8 GB of RAM) are shown in Table 1. The data shows the elapsed-time (s) for the two sequential phases of the visualization process. The former is the construction phase, which shows the time spent for converting SMILES notations into numeric multi-dimensional representations. The latter is the reduction phase, which shows the time spent for reducing the

multi-dimensions to 2D. We separately executed the reduction phase for the three different algorithms of PCA, t-SNE, and UMAP. The construction phase of the structural similarity is linearly correlated with the size of the dataset and it is about three to six times faster than the tailored similarity. For the largest dataset shown in Table 1, the construction time is approximately 74 and 453 s for structural and tailored similarities, respectively. In the reduction phase, the elapsed-time for PCA is linearly correlated with the size of the dataset, whereas it is polynomially correlated for UMAP and t-SNE. For the largest dataset, the reduction time using structural similarity was approximately 19, 180, and 429 s for PCA, UMAP, and t-SNE.

Distance Property Relationship Comparison

In this set of experiments, we compare the performances of various configurations of chemical space visualization on dealing with the APC problem. While traditional methods like Structure-Activity Relationships (SAR)²⁸ and Structure-Activity Landscape (SAL)²⁹ indexes provide quantification and detection of APCs of molecular datasets, there is no available method for quantifying APCs for chemical space visualizations. To measure the activity/property difference of similar (i.e. close) molecules in the visualized chemical space, we use the herein introduced metric of the DPR scores. Unlike SAR and SAL indexes, the DPR score defines the similarity based on distances of molecules on the reduced chemical space as opposed to basing it on their common chemical substructures. To achieve this, the DPR algorithm first calculates the Euclidean distances between all pairs of molecules and sorts them in ascending order (i.e. closest first). Next, for a selected top percentage, an average of the property difference is calculated. Table 2 shows the top 1, 2, and 5% DPR scores of the chemical space visualizations that are obtained by using various experimental configurations. On all datasets, among the three dimensionality reduction methods, t-SNE method produced the best DPR scores, while UMAP performed better than PCA. In addition, for the majority of experiments, the tailored similarity showed a better performance than the structural similarity.

Web Application

The ChemPlot web application provides users with an easy-to-use application programming interface (API) for visualizing chemical spaces. Users can upload their molecular datasets and choose the available visualization options from the left panel. The API supports main functionalities of ChemPlot, including the similarity methods, dimensionality reduction methods, and plotting types. Additionally, it can be used to remove outliers and set a random state for reproducible plots. It generates interactive plots and allows users to export their visualized chemical spaces as an image or HTML file. Figure 5 shows an example view of the web application, which is openly accessible at <https://www.amdlab.nl/chemplot/>.

DISCUSSION

In this study, we introduced ChemPlot, a Python library for chemical space visualization. When designing ChemPlot, we aimed to provide a straightforward library where coding would be smooth and intuitive. This was achieved by providing a simple code flow and naming, handling the function parameters automatically, and preparing a descriptive user manual. ChemPlot allows users to convert molecular data files into chemical space visualizations simply by using a few lines of code and without setting any input parameters. Moreover, we developed a web application to streamline the visualization of chemical spaces without the

Dataset	PCA						t-SNE						UMAP					
	1%		2%		5%		1%		2%		5%		1%		2%		5%	
	S	T	S	T	S	T	S	T	S	T	S	T	S	T	S	T	S	T
FreeSolv ²⁵	2.27	2.23	2.63	2.42	2.85	2.63	1.63	1.46	2.09	1.74	2.75	2.26	2.24	1.60	2.65	1.90	3.23	2.43
Huuskonen ²⁶	1.77	1.10	1.81	1.18	1.86	1.31	1.39	0.97	1.62	1.10	1.84	1.27	1.52	1.00	1.65	1.15	1.86	1.33
BACE ²¹	1.03	1.20	1.07	1.25	1.18	1.29	0.81	0.89	0.90	1.01	1.06	1.18	0.82	0.92	0.89	1.06	1.11	1.25
Lipophilicity ²²	1.29	1.26	1.30	1.26	1.32	1.27	1.16	1.12	1.26	1.18	1.33	1.24	1.25	1.15	1.32	1.19	1.38	1.23
AqSolDB ²⁴	2.24	1.91	2.25	1.95	2.28	2.01	2.11	1.69	2.25	1.85	2.39	1.98	2.12	1.80	2.19	1.93	2.30	2.08
ClinTox ²²	0.11	0.11	0.11	0.12	0.11	0.12	0.10	0.11	0.12	0.12	0.13	0.12	0.13	0.12	0.14	0.13	0.15	0.12
BBBP ²³	0.23	0.22	0.23	0.22	0.23	0.23	0.17	0.19	0.20	0.21	0.24	0.26	0.20	0.19	0.22	0.22	0.24	0.25

Table 2. Top 1, 2, and 5% DPR scores of ChemPlot for the various experimental configurations applied on the sample datasets. S and T denote the structural and tailored similarity methods, respectively.

need of a coding background and essentially for daily use. Thus, the ChemPlot Python library alongside its user-friendly web application are designed to serve a broad community of users.

A noteworthy feature of ChemPlot is the application of tailored similarity for the visualization of chemical spaces. Before discussing this similarity feature, it is important to understand the similarity concept. Similarity is relative and it depends on the target and the population, therefore there is no globally applicable definition of similarity¹¹. The traditionally used form is the structural similarity that attempts to evaluate the similarity using all possible aspects of the given molecular datasets and without taking the target property into account. This approach is truly suitable when one wants to have a general overview of the dataset³⁰. However, structural similarity often fails when searching for new molecules with target properties. For example, when searching for molecules with a desired aqueous solubility value, one can expect that the neighboring molecules on the visualized chemical space would have close solubility values. Neighboring molecules, however, are more likely to have closer solubility values when, instead of the similarity defined by all the features, the similarity is defined essentially by the features that are known to affect the solubility. In relation to this, ChemPlot includes the tailored similarity option, which automatically identifies the correlated features with the given target and visualizes the chemical space based on them. The chemical spaces that are visualized by using the tailored similarity are less likely to be affected by the APC issue. Inline with this, for the majority of the datasets shown in Table 2, the tailored similarity produced lower DPR scores than the structural similarity.

In addition to its use in high-throughput virtual screening, diversity analysis, and virtual library design, it is also possible to use ChemPlot for visualizing the applicability domain of AI models. Here, the applicability domain refers to the region of chemical space where a model is expected to perform reliably and make confident predictions. For instance, when the predicted molecule is within the applicability domain, confidence can be put in the prediction, and when it is outside the applicability domain the prediction can be considered as dubious. By visualizing the chemical space of the training and test datasets of the models, the level of confidence of the prediction for new molecules can be determined based on their chemical space placement. Additionally, ChemPlot can also be used to verify whether the dataset is properly split into train and test sets that are both adequately covering the chemical space^{31,32}. Furthermore, by employing the clustering feature of ChemPlot, a test set region that is not properly covered by the training set can be intentionally put aside in order to evaluate the extrapolation capability of the model. Figure 4 shows an example use case of ChemPlot for identifying and isolating a test set that lies outside of the

domain of the training set. This way, the performance of a ML model, which has been trained with the data from the orange region, on the data from the blue region can be used as an indicator of the extrapolation capability of the model. Accordingly, we expect that ChemPlot will serve as a complementary tool to the ML models that are increasingly applied in the molecular informatics studies.

Advancing research software sustainability is a dire challenge. Software released as desktop or web applications can suffer from a lack of maintenance, therefore over time, can become inaccessible or completely unusable. We released ChemPlot as an open-source library on GitHub, thereby enabling the community to contribute to its future development. Code developers may contribute, such as by adding new features pertaining to dimensionality reduction techniques or similarity methods. Additionally, they may have a chance to resolve user-specific technical issues when encountered. Moreover, in ChemPlot, an automated unit test workflow feature is implemented to validate the integrity of the to-be developed features by contributors in future (see Methods). Furthermore, we developed a distribution workflow that automatically builds and distributes the new ChemPlot package when a new version is released. To pave the way for long-term software quality, we share the responsibility in maintenance and development of ChemPlot. Thus, it can dynamically evolve in parallel to the future needs of the community.

The most significant practical limitation of ChemPlot is the computation time that will be required for excessively large datasets. In our performance tests, we encountered bottlenecks in both the construction and the reduction phases. In the construction phase, the calculation time of fingerprints and descriptors depends on both the total number of molecules and the total number of atoms in molecules. Also, the tailored similarity feature requires an additional step of feature selection. ChemPlot uses the least absolute shrinkage and selection operator (lasso) regression analysis method and the logistic regression algorithms for feature selection. Depending on the algorithm used in this step, additional computation time is required. Therefore, the construction phase takes more time for the tailored similarity. To accelerate this process, in the future versions of ChemPlot, the selection algorithms for instance can be optimized or replaced entirely by faster methods for the large datasets. In the reduction phase, the production of all the plots took less than three minutes for both PCA and UMAP methods. However, for t-SNE, the production of the plots of large datasets took significantly longer time. For instance, for the largest dataset we tested (41,127 instances), it took approximately seven minutes to generate the plot by t-SNE (see Table 1). We used Barnes-Hut³³ implementation from scikit-learn³⁴ library which runs in $O(N \log N)$, where N is the number of instances. The reduction time is naturally also affected by the number of dimensions that will be reduced. Therefore, it is usually recommended to use t-SNE once after the dimensions are reduced by PCA to a fixed value^{33,35} although there is the risk of information loss. ChemPlot includes a PCA pre-reduction option but the performance tests that are shown in Table 1 were conducted without this pre-reduction step. Therefore, for t-SNE, the total time consumed is largely penalized by the number of dimensions as well. To increase t-SNE's processing rate, it may be worthwhile to implement its recently proposed variants³⁵⁻³⁸ in ChemPlot in future.

METHODS

Structural Similarity

When computing the chemical space visualization based on the structural similarity, the list of molecules is converted into Extended-Connectivity Fingerprints (ECFPs)³⁹. ECFPs are bit-vectors where each bit represents the presence or absence of a particular substructure. Substructures are extracted from the main structure by starting from each non-hydrogen atom and extending to the neighbor atoms until a specified distance is reached. Extracted substructures are hashed and mapped into fixed-sized bit-vector. ChemPlot uses the RDKit¹ library to convert SMILES and InChI notations into ECFPs with a bit-vector length of 2,048 bits and radius of 2 adjacent atoms. After each molecule is converted, for all molecules, the bits that contain only 0s or only 1s are removed from the bit-vectors. The remaining number of bits determines the total number of dimensions and they are used as the input for the dimensionality reduction phase.

Tailored Similarity

When computing the chemical space visualization based on the tailored similarity, the list of molecules is converted into a set of descriptors as computed by using the Mordred library³. Initially, a total of 201 physico-chemical descriptors are calculated. A molecule is removed from the dataset in the case when a descriptor could not have been successfully computed. The list of descriptors is then used to form a matrix, in which the rows represent compounds and the columns represent descriptors. Next, the descriptors that are correlated with the target property are selected by using lasso regression for the numerical target values or by using logistic regression for the categorical target values. Lasso regression uses $L1$ regularization with 0.05 of α (regularization multiplier) and has a maximum iteration of 10,000. Logistic regression uses $L1$ regularization with 0.3 of C (inverse of regularization strength) and *liblinear* as the optimization method. The resulting matrix of the selected descriptors is used as the input for the dimensionality reduction phase.

Principal Component Analysis

PCA⁴ is a linear dimensionality reduction algorithm, which projects the data points onto principal components by maximizing the variance. ChemPlot applies PCA from scikit-learn³⁴ library with the default parameters. The two most significant principal components are used as the reduced dimensions (axes of the graph) in the visualization step.

t-distributed Stochastic Neighbor Embedding

t-SNE⁷ is a non-linear dimensionality reduction algorithm that converts the similarities between data points into joint probabilities. It then minimizes the difference between the joint probability distributions of the high-dimensional data and the low-dimensional embedding. It is a stochastic process that produces different results from different initialization parameters. Except the *perplexity* parameter, ChemPlot applies t-SNE from scikit-learn³⁴ library using its default parameters. The *perplexity* parameter is computed automatically by the pre-trained model as described below.

Uniform Manifold Approximation and Projection

UMAP¹⁰ is a non-linear dimensionality reduction algorithm that constructs a particular weighted k-neighbor graph for the given data points and then computes a low dimensional layout of the graph. It is based on a stochastic process that produces different results from different initialization parameters. ChemPlot applies UMAP with the default parameters provided by the UMAP library, except the *n_neighbors* and *min_dist* that are computed automatically by the pre-trained model described below.

Distance Property Relationship Score

DPR score is a new method we developed to quantify the activity/property difference of similar molecules. It calculates the average activity/property difference of the most similar molecule pairs for a given percentage. An example pseudo-code is given below.

```
function DPR_score(Argument1: data, Argument2: P)
    visualize the chemical space of data
    get coordinates of each molecule
For each molecule couple:
        calculate their distance
        calculate their activity/property difference
    sort the couples based on the distance (ascending)
    select the top P percent
return the average absolute activity/property difference of top P
```

Automatic Parameter Assignment

The most important parameters that affect the visualization are *perplexity* for t-SNE and *number of neighbors* for UMAP. However, it is not possible to set fixed values of these parameters that would work best for all datasets. In order to determine the best values for a given dataset, we developed a linear model. For this purpose, we used 20 molecular datasets of varying sizes and generated multiple plots for each dataset by assigning different values to the target parameters. The best plots for human perception and the respective values were selected by two users for each dataset based on the clearness and the balanced distribution of the clusters and data points. The selected values were used as the labels to form linear equations between the data size and the target parameters. By using the trained model, ChemPlot automatically assigns the parameters to generate visually meaningful plots.

Clustering

For clustering of the molecules in a reduced chemical space, ChemPlot uses the *k*-means⁴⁰ method as implemented in the scikit-learn³⁴ library. The default value for the number of clusters is "5", while all other parameters are taken as are from scikit-learn. Once clustering of the data in the reduced dimensions is complete, it is possible to highlight a selected cluster by passing a list of cluster ids when calling the *visualize_plot* function.

Static Plot

ChemPlot uses the Seaborn⁴¹ library for generating the static plots. Static plots can be created by using one of three different visualization options, including the scatter, kernel density, and hexagonal bin plots. Static plots can be exported in PNG, JPEG, PDF, and SVG file formats.

Interactive Plot

ChemPlot uses the Boken⁴² library for generating the interactive plots. The interactive plot allows users to interact actively with the visualization by providing drag, highlight, zoom, save, and reset functions. Hovering over the data points displays a tool-tip that contains a basic image of the molecule as rendered by RDKit¹. The interactive plot can be exported in HTML format.

Removing Outliers

ChemPlot identifies the molecules as outliers in the case when their $|z\text{-score}| \geq 3$. Z-scores are calculated by using SciPy⁴³ library. The molecules that are identified as outliers can be removed from the plots by setting the `remove_outliers` parameter to "True".

Unit Tests

Chemplot uses `unittest` Python framework for automated unit testing. Unit tests cover all construction, dimensionality reduction, and visualization methods. For all test cases, the related explanatory text is also provided. Unit tests are executed on three types of datasets: i) a dataset with a numerical target, ii) a dataset with a categorical target, and iii) a dataset that contains erroneous SMILES representations.

Visualization Tests

Visualization tests only cover the static plots. The test generates plots for the selected datasets with all possible combinations of parameters and puts them into a single PDF file for human inspection. It does not contain any automated validation.

Performance Tests

Performance tests measure the elapsed-time for the construction and the dimension reduction phases. The test is executed for nine molecular datasets of different sizes and by using all the similarity and dimension reduction methods. Each performance test provides two output files, where one includes a table of execution times shown in s and the other contains meta information of the test environment.

Web Application

ChemPlot web application is designed as an independent open-source software that imports ChemPlot as a library. It uses Streamlit library for creating the web interface.

Code Availability

The source code of ChemPlot and the web application is available on GitHub repositories, respectively (<https://github.com/mcsorkun/ChemPlot>, <https://github.com/mcsorkun/ChemPlot-web>). The user manual is accessible via <https://chemplot.readthedocs.io/>. The collective repositories of ChemPlot can be found at AMD group GitHub account on <https://github.com/ergroup/ChemPlot>.

References

1. Rdkit: Cheminformatics and machine learning software (2013).
2. Yap, C. W. Padel-descriptor: An open source software to calculate molecular descriptors and fingerprints. *J. Comput. Chem.* **32**, 1466–1474 (2011).
3. Moriwaki, H., Tian, Y.-S., Kawashita, N. & Takagi, T. Mordred: a molecular descriptor calculator. *J. Cheminformatics* **10**, 1–14 (2018).
4. Wold, S., Esbensen, K. & Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **2**, 37–52 (1987).
5. Kohonen, T. The self-organizing map. *Proc. IEEE* **78**, 1464–1480 (1990).
6. Agraftotis, D. K., Rassokhin, D. N. & Lobanov, V. S. Multidimensional scaling and visualization of large molecular similarity tables. *J. Comput. Chem.* **22**, 488–500 (2001).
7. Van der Maaten, L. & Hinton, G. Visualizing data using t-sne. *J. Mach. Learn. Res.* **9** (2008).

8. Probst, D. & Reymond, J.-L. Visualization of very large high-dimensional data sets as minimum spanning trees. *J. Cheminformatics* **12**, 1–13 (2020).
9. Maggiora, G. M. & Bajorath, J. Chemical space networks: a powerful new paradigm for the description of chemical space. *J. Comput. Mol. Des.* **28**, 795–802 (2014).
10. McInnes, L., Healy, J. & Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint <https://arxiv.org/abs/1802.03426>* (2018).
11. Maggiora, G., Vogt, M., Stumpfe, D. & Bajorath, J. Molecular similarity in medicinal chemistry: miniperspective. *J. Medicinal Chem.* **57**, 3186–3204 (2014).
12. Gütlein, M., Karwath, A. & Kramer, S. Ches-mapper-chemical space mapping and visualization in 3d. *J. Cheminformatics* **4**, 1–16 (2012).
13. Sander, T., Freyss, J., von Korff, M. & Rufener, C. Datawarrior: an open-source program for chemistry aware data visualization and analysis. *J. Chem. Inf. Model.* **55**, 460–473 (2015).
14. Gonzalez-Medina, M. & Medina-Franco, J. L. Platform for unified molecular analysis: Puma. *J. Chem. Inf. Model.* **57**, 1735–1740 (2017).
15. Medina-Franco, J. L., Sánchez-Cruz, N., López-López, E. & Díaz-Eufracio, B. I. Progress on open chemoinformatic tools for expanding and exploring the chemical space. *J. Comput. Mol. Des.* 1–14 (2021).
16. Takahashi, Y., Konji, M. & Fujishima, S. Molspace: a computer desktop tool for visualization of massive molecular data. *J. Mol. Graph. Model.* **21**, 333–339 (2003).
17. Škoda, P. & Hoksza, D. Chemical space visualization using viframe. In *2013 IEEE/ACIS 12th International Conference on Computer and Information Science (ICIS)*, 541–546 (IEEE, 2013).
18. Awale, M., van Deursen, R. & Reymond, J.-L. Mqn-mapplet: Visualization of chemical space with interactive maps of drugbank, chembl, pubchem, gdb-11, and gdb-13. *J. Chem. Inf. Model.* **53**, 509–518 (2013).
19. Awale, M., Probst, D. & Reymond, J.-L. Webmolcs: a web-based interface for visualizing molecules in three-dimensional chemical spaces. *J. Chem. Inf. Model.* **57**, 643–649 (2017).
20. Gute, B. D., Basak, S. C., Mills, D. & Hawkins, D. M. Tailored similarity spaces for the prediction of physicochemical properties. *Internet Electron. J. Mol. Des.* **1**, 374–387 (2002).
21. Subramanian, G., Ramsundar, B., Pande, V. & Denny, R. A. Computational modeling of β -secretase 1 (bace-1) inhibitors using ligand based approaches. *J. Chem. Inf. Model.* **56**, 1936–1949 (2016).
22. Wu, Z. *et al.* Moleculenet: a benchmark for molecular machine learning. *Chem. Sci.* **9**, 513–530 (2018).
23. Martins, I. F., Teixeira, A. L., Pinheiro, L. & Falcao, A. O. A bayesian approach to in silico blood-brain barrier penetration modeling. *J. Chem. Inf. Model.* **52**, 1686–1697 (2012).
24. Sorkun, M. C., Khetan, A. & Er, S. Aqsolddb, a curated reference set of aqueous solubility and 2d descriptors for a diverse set of compounds. *Sci. Data* **6**, 1–8 (2019).
25. Mobley, D. L. & Guthrie, J. P. Freesolv: a database of experimental and calculated hydration free energies, with input files. *J. Comput. Mol. Des.* **28**, 711–720 (2014).

26. Huuskonen, J. Estimation of aqueous solubility for a diverse set of organic compounds based on molecular topology. *J. Chem. Inf. Comput. Sci.* **40**, 773–777 (2000).
27. Coveralls. <https://coveralls.io/github/mcsorkun/ChemPlot>. [Online; accessed 2022-01-21].
28. Peltason, L. & Bajorath, J. Sar index: quantifying the nature of structure- activity relationships. *J. Medicinal Chem.* **50**, 5571–5578 (2007).
29. Guha, R. & Van Drie, J. H. Structure- activity landscape index: identifying and quantifying activity cliffs. *J. Chem. Inf. Model.* **48**, 646–658 (2008).
30. Sorkun, E., Zhang, Q., Khetan, A., Sorkun, M. C. & Er, S. Reddb, a computational database of electroactive molecules for aqueous redox flow batteries. *ChemRxiv preprint* [10.26434/chemrxiv.14398067.v1](https://doi.org/10.26434/chemrxiv.14398067.v1) (2021).
31. Sorkun, M. C., Koelman, J. V. A. & Er, S. Pushing the limits of solubility prediction via quality-oriented data selection. *Isience* **24**, 101961 (2021).
32. Redpred: Redox energy prediction tool for redox flow battery molecules. <https://github.com/mcsorkun/RedPred/> (2022).
33. Van Der Maaten, L. Accelerating t-sne using tree-based algorithms. *The J. Mach. Learn. Res.* **15**, 3221–3245 (2014).
34. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
35. Linderman, G. C., Rachh, M., Hoskins, J. G., Steinerberger, S. & Kluger, Y. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nat. Methods* **16**, 243–245 (2019).
36. Ulyanov, D. Multicore-tsne. <https://github.com/DmitryUlyanov/Multicore-TSNE> (2016).
37. Chan, D. M., Rao, R., Huang, F. & Canny, J. F. t-sne-cuda: Gpu-accelerated t-sne and its applications to modern data. In *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 330–338 (IEEE, 2018).
38. Fujiwara, Y., Ida, Y., Kanai, S., Kumagai, A. & Ueda, N. Fast similarity computation for t-sne. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 1691–1702 (IEEE, 2021).
39. Rogers, D. & Hahn, M. Extended-connectivity fingerprints. *J. Chem. Inf. Model.* **50**, 742–754 (2010).
40. Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory* **28**, 129–137 (1982).
41. Waskom, M. L. seaborn: statistical data visualization. *J. Open Source Softw.* **6**, 3021 (2021).
42. Bokeh Development Team. *Bokeh: Python library for interactive visualization* (2018).
43. Virtanen, P. *et al.* Scipy 1.0: fundamental algorithms for scientific computing in python. *Nat. Methods* **17**, 261–272 (2020).

Acknowledgements

The authors acknowledge funding from the initiative “Computational Sciences for Energy Research” of Shell and the Dutch Research Council (NWO) grant no 15CSTT05. This work was sponsored by NWO Exact and Natural Sciences for the use of supercomputer facilities.

Author contributions

M.C.S. and D.M. developed the codes for ChemPlot and its web application. S.E. supervised the project. All authors contributed to the writing of the manuscript.

Competing interests

The authors declare no competing interests.