NAME

VisualizeChemspaceUsingTMAP.py - Visualize chemspace

SYNOPSIS

VisualizeChemspaceUsingTMAP.py [--categoricalDataCols <collabel1,... or colnum1,...>] [--categoricalDataColormaps <Colormap1, Colormap2,...>] [--categoricalDataMaxDisplay <number>] [--colmode <collabel or colnum>] [--colSMILES <text or number>] [--faerunConfigParams <Name,Value,...>] [--faerunScatterPlotParams <Name,Value,...>] [--infileDelimiter <comma, tab, or space>] [--lshForestFileWrite <yes or no>] [--lshForestFileRestore <yes or no>] [--lshForestParams <Name,Value,...>] [--lshLayoutConfigParams <Name,Value,...>] [--mergeHTMLandJSFiles <yes or no>] [--minHashFPParams <Name,Value,...>] [--mp <yes or no>] [--mpParams <Name,Value,...>] [--numericalDataCols <collabel1,... or colnum1,...>] [--numericalDataColormaps <Colormap1, Colormap2,...>] [--overwrite] [--quiet <yes or no>] [--structureDisplayDataCols <collabel1,... or colnum1,...>] [--tmapDisplayMsg <text>] [-w <dir>] -i <infile> -o <outfile>

VisualizeChemspaceUsingTMAP.py -h | --help | -e | --examples

DESCRIPTION

Generate an interactive TreeMAP (TMAP) [Ref 171, 172] visualization for molecules in a text input file. The text input file must have a column containing SMILES strings. In addition, it must contain at least one column corresponding to categorical or numerical data for coloring TMAP nodes. You may optionally map multiple categorical and numerical data columns on to a TMAP visualization. A HTML file is generated for interactive visualization of chemspace in a browser.

The TMAP methodology is able to generate a reasonably interactive visualization for relatively large data sets. A brief description of the methodology is as follows. A set of MinHash Fingerprints (MHFPs) are calculated for molecules in input file followed by the generation of a Locality Sensitivity Hashing (LSH) forest employing MHFPs. A c-approximate k-Nearest Neighbor Graph (c-k-NNG) is constructed from LSH, which is used to construct a Minimum Spanning Tree (MST) or Forest (MSF). The final TMAP visualization is generated by laying out MST and MSF on a plane using an algorithm provided by the Open Graph Drawing Framework (OGDF). The OGDF provides flexibility to adjust graph layout methodology in terms of not only aesthetics but also computational time.

The supported input file formats are: CSV (.csv) TSV (.txt or .tsv), SMILES (.smi)

The supported output file format is: HTML (.html).

OPTIONS

--categoricalDataCols <collabel1,... or colnum1,...> [default: none]

A comma delimited list of column labels or numbers corresponding to categorical data to map on a TMAP visualization.

--categoricalDataColormaps <Colormap1, Colormap2,...> [default: auto]

A comma delimited list of color map names corresponding to categorical data. The default is to use 'tab10' color map name for mapping categorical data on a TMAP. The number of specified color maps must match the number of categorical data columns. You must specify valid color map names supported by Matplotlib. No validation is performed. Example color map names for categorical data: Pastel1, Pastel2, Paired, Accent, Dark2, Set1, Set2, Set3, tab10, tab20, tab20b, tab20c.

--categoricalDataMaxDisplay <number> [default: 6]

Maximum number of categories in a category column to display on a TMAP visualization. The rest of the categories are aggregated under a new category named 'Other' before mapping on to a TMAP visualization.

-c, --colmode <collabel or colnum> [default: collabel]

Use column number or name for the specification of columns in input text file containing SMILES strings and molecule names along with any categorical or numerical data.

--colSMILES <text or number> [default: auto]

Column name or number corresponding to SMILES strings. The default value is automatically set based on the value of '-c, --colmode': 'SMILES' for 'collabel'; SMILES string column number for 'colnum'. SMILES strings must be present in input file.

-e, --examples

Print examples.

--faerunConfigParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for configuring faerun (Ref 172) to generate a TMAP visualization.

The supported parameter names along with their default and possible values are shown below:

```
clearColor, #000000
showLegend, yes  [ Possible values: yes or no ]
legendTitle, Legend
legendOrientation, vertical  [ Possible values: vertical or
    horizontal ]
legendNumberFormat, {:.2f}
scale, 750.0
alphaBlending, no  [ Possible values: yes or no ]
antiAliasing, yes  [Possible values: yes or no]
thumbnailWidth, 250
thumbnailFixed, no  [ Possible values: yes or no ]
```

A brief description of parameters, as available in the code for faerun, is provided below:

```
clearColor: Background color
showLegend: Show legend at lower right
legendTitle: Legend title
legendOrientation: Legend Orientation
legendNumberFormat: Number string format applied to numbers
    displayed in legend
scale: Scaling factor for scaling normalized coordinates
AlphaBlending: Activate alpha blending. It is required for smoothCircle
    shader.
antiAliasing: Activate anti-aliasing. It might adversly impact
    rendering performance.
thumbnailWidth: Width of thumbnail images for structures
thumbnailFixed:  Show thumbnail images at a fixed location at the
    top instead of next to the mouse
```

--faerunScatterPlotParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for generating scatter plot representing a TMAP using faerun (Ref 172).

The supported parameter names along with their default and possible values are shown below:

```
shader, circle  [ Possible values: circle, smoothCircle,
    sphere, or any valid value]
pointScale, auto  [ 4 if MolCout<=10K; 2 if MolCount<=100K; else 1 ]
maxPointSize, 100.0
fogIntensity, 0.0
interactive, yes  [ Possible values: yes or no ]
```

A brief description of parameters is provided below:

```
shader: Shader to use for visualizating data points
pointScale: Relative size of data points
maxPointSize: Maximum size of the data points during zooming
fogIntensity: Intensity of distance fog
interactive: Generate interactive scatter plot
```

-h, --help

Print this help message.

-i, --infile <infile>

Input file name. The SMILES strings must be present in the input file. Supported formats: CSV (.csv) TSV (.txt or .tsv), or SMILES (.smi)

--infileDelimiter <comma, tab, or space> [default: auto]

Input file delimiter for processing data. The default value is automatically set based on the type of input file: comma - CSV (.csv); tab - TSV (.txt or .tsv); space - SMILES (.smi)

--lshForestFileWrite <yes or no> [default: yes]

Write LSH forest data a file for subsequent generation of a TMAP visualization. Default file name: <OutfileRoot>_LSHForest.dat. The LSH forest data is generated using MinHash fingerprints. You may restore LSH forest data using '--lshForestFileRestore' option to skip the generation of fingerprints.

--lshForestFileRestore <yes or no> [default: no]

Check and restore LSH forest data from a file for generating a TMAP visualization and skip the generation of MinHash fingerprints. Default file name: <OutfileRoot>_LSHForest.dat

--lshForestParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for generating LSH (Locality Sensitivity Hashing) forest from MinHash fingerprints.

The supported parameter names along with their default and possible values are shown below:

```
dim, 2048
numPrefixTrees, auto  [ 128 if MolCount <= 10K else 8 ]
store, yes  [ Possible values: yes or no ]
```

A brief description of parameters, as available in the code for LSH, is provided below:

```
dim: Dimensionality of MinHashes to be added to LSHForest
numPrefixTrees: Number of prefix trees to use
store: store the data for enhanced retrieval
```

--lshLayoutConfigParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for configuring LSH (Locality Sensitivity Hashing) layout.

The supported parameter names along with their default and possible values are shown below:

```
k, auto  [ 75 if MolCount <= 10K else 10]
kc, auto  [ 20 if MolCount <= 10K else 10]
fmeIterations, 1000
fmeRandomize, no  [ Possible values: yes or no ]
fmeThreads, 4
fmePrecision, 4
slRrepeats, auto  [ 2 if MolCount <= 10K else 1]
slExtraScalingSteps, auto  [ 4 if MolCount <= 10K else 2 ]
slScalingMin, 1.0
slScalingMax, 1.0
slScalingType, RelativeToDrawing  [ Possible values: Absolute,
    RelativeToAvgLength, RelativeToDesiredLength, or
    RelativeToDrawing ]
mmmRepeats, auto  [ 2 MolCount <= 10K else 1 ]
placer, Barycenter  [ Possible valeues: Barycenter, Solar, Circle,
    Median, Random, or Zero ]
merger, LocalBiconnected  [ Possible values: EdgeCover,
    LocalBiconnected, Solar, or IndependentSet ]
mergerFactor, 2.0
mergerAdjustment, 0
nodeSizeDenominator, auto  [ 65 if MolCout <= 10K else 70.0]
```

A brief description of parameters, as available in the code for LSH, is provided below:

```
k: Number of nearest neighbors used to create k-nearest neighbor
    graph
kc: Scalar by which k is multiplied before querying LSH forest.
    The results are then sorted in decreasing order based on linear
    scan distances.
fmeIterations: Maximum number of iterations of Fast Multipole
```

```
        Embedder (FME)
    fmeRandomize: Randomize FME layout at the start
    fmeThreads: Number of threads for FME
    fmePrecision: Number of coefficients of multipole expansion
    slRepeats: Number of repeats of scaling layout algorithm
    slExtraScalingSteps: Number of repeats of scaling
    slScalingMin: Minimum scaling factor
    slScalingMax: Maximum scaling factor.
    slScalingType: Scaling type corresponding to relative scale of graph
    mmmRepeats, Number of repeats of layout at each level
    placer: Methodology for defining initial positions of vertices in a
        graph at each level
    merger: Vertex merging methodology used during coarsening phase
        of multilevel algorithm
    mergerFactor: Ratio of sizes between two levels up to which merging
        is performed.  It doesn't apply to all merging methodologies.
    mergerAdjustment: Edge  length  adjustment  for merging methodology.
        It doesn't apply to all merging methodologies.
    nodeSizeDenominator: Node size denominator affecting the magnitude
        of repelling force between nodes. Node size corresponds to
        1.0 / nodeSizeDenominator. You may want to increase the value
        nodeSizeDenominator to decrease node size and resolve overlaps
        in  a crowded tree.
```

--mergeHTMLandJSFiles <yes or no> [default: yes]

Merge TMAP JS data file into HTML file and delete JS data file. Default file names: <OutfileRoot>.html, <OutfileRoot>.js.

--minHashFPParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for generating Min Hash Fingerprints (MHFP).

The supported parameter names along with their default and possible values are shown below:

```
    radius, 3
    rings, yes  [ Possible values: yes or no ]
    kekulize, yes  [ Possible values: yes or no ]
    sanitize, yes  [ Possible values: yes or no ]
    minRadius, 1
    numPermutations, 2048
    seed, 42
```

A brief description of parameters, as available in the code for MHFP, is provided below:

```
    radius:  MHFP radius (A radius of 3 corresponds to MHFP6)
    rings:  Include rings in shingling
    kekulize:  Kekulize SMILES
    sanitize:  Sanitize SMILES
    minRadius: Minimum radius that is used to extract n-grams
    numPermutations: Number of permutations used for hashing
    seed: Random number seed for numpy.random
```

--mp <yes or no> [default: no]

Use multiprocessing for the generation of fingerprints.

By default, input data is retrieved in a lazy manner via mp.Pool.imap() function employing lazy RDKit data iterable. This allows processing of arbitrary large data sets without any additional requirements memory.

All input data may be optionally loaded into memory by mp.Pool.map() before starting worker processes in a process pool by setting the value of 'inputDataMode' to 'InMemory' in '--mpParams' option.

A word to the wise: The default 'chunkSize' value of 1 during 'Lazy' input data mode may adversely impact the performance. The '--mpParams' section provides additional information to tune the value of 'chunkSize'.

--mpParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs to configure multiprocessing during the

generation of fingerprints.

The supported parameter names along with their default and possible values are shown below:

```
chunkSize, auto
inputDataMode, Lazy   [ Possible values: InMemory or Lazy ]
numProcesses, auto   [ Default: mp.cpu_count() ]
```

These parameters are used by the following functions to configure and control the behavior of multiprocessing: mp.Pool(), mp.Pool.map(), and mp.Pool.imap().

The chunkSize determines chunks of input data passed to each worker process in a process pool by mp.Pool.map() and mp.Pool.imap() functions. The default value of chunkSize is dependent on the value of 'inputDataMode'.

The mp.Pool.map() function, invoked during 'InMemory' input data mode, automatically converts RDKit data iterable into a list, loads all data into memory, and calculates the default chunkSize using the following method as shown in its code:

```
chunkSize, extra = divmod(len(dataIterable), len(numProcesses) * 4)
if extra: chunkSize += 1
```

For example, the default chunkSize will be 7 for a pool of 4 worker processes and 100 data items.

The mp.Pool.imap() function, invoked during 'Lazy' input data mode, employs 'lazy' RDKit data iterable to retrieve data as needed, without loading all the data into memory. Consequently, the size of input data is not known a priori. It's not possible to estimate an optimal value for the chunkSize. The default chunkSize is set to 1.

The default value for the chunkSize during 'Lazy' data mode may adversely impact the performance due to the overhead associated with exchanging small chunks of data. It is generally a good idea to explicitly set chunkSize to a larger value during 'Lazy' input data mode, based on the size of your input data and number of processes in the process pool.

The mp.Pool.map() function waits for all worker processes to process all the data and return the results. The mp.Pool.imap() function, however, returns the the results obtained from worker processes as soon as the results become available for specified chunks of data.

The order of data in the results returned by both mp.Pool.map() and mp.Pool.imap() functions always corresponds to the input data.

--numericalDataCols <collabel1,... or colnum1,...> [default: none]

A comma demlimited list of column labels or numbers corresponding to numerical data to map on a TMAP visualization.

--numericalDataColormaps <Colormap1, Colormap2,...> [default: auto]

A comma demlimited list of color map names corresponding to numerical data. The default is to use 'viridis' color map name for mapping numerical data on a TMAP. The number of specified color maps must mtach the number of numerical data columns. You must specify valid color map names supported by Matplotlib. No validation is performed. Example color map names for numerical data: viridis, plasma, inferno, magma, cividis.

-o, --outfile <outfile>

Output HTML file name for writing out a TMAP visualization.

--overwrite

Overwrite existing files.

-q, --quiet <yes or no> [default: no]

Use quiet mode. The warning and information messages will not be printed.

--structureDisplayDataCols <collabel1,... or colnum1,...> [default: auto]

A comma delimited list of column labels or numbers corresponding to data to display under a thumbnail image of a structure in a TMAP visualization. The default column is set to 'Name' and it is automatically shown. In addition, the SMILES string column is always used to display SMILES under the structures.

-t, --tmapDisplayMsg <text> [default: auto]

A brief message to display at the top left in HTML page containing a TMAP visualization. You must specify a

valid HTML string. No validation is performed. Default message: TMAP chemspace visualization
 Input file: <InfileName>
Number of molecules: <Count>

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To visualize chemspace for SMILES strings present in a column name SMILES in input file, mapping a categorical data column on TMAP, writing out LSH forest for subsequent use to skip the generation of fingerprints, merging TMAP JS file into HTML file, and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols Source
  -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example for SMILES strings in column name SMILES in input file and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --colSMILES SMILES
  --categoricalDataCols Source
  -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example for mapping categrorical data in column number 4 in input file and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --colmode colnum
  --categoricalDataCols 4
  -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example for mapping both categrorical and numerical data coumns and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols "Source"
  --numericalDataCols "MolWt,MolLogP"
  -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example for mapping both categrorical and numerical data coumns along with specified colormaps and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols "Source"
  --categoricalDataColormaps "tab10"
  --numericalDataCols "MolWt,MolLogP"
  --numericalDataColormaps "viridis, plasma"
  -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example for mapping both categrorical and numerical data coumns along with displaying specific data under the structure display and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols "Source"
  --numericalDataCols "MolWt,NHOHCount,NOCount,MolLogP,
  NumRotatableBonds,TPSA" --structureDisplayDataCols "Name,ID"
  -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example for restoring LSH forest data from a file to skip the generation of fingerpritns and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols Source
   --lshForestFileRestore yes -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example in multiprocessing mode on all available CPUs without loading all data into memory and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols Source
```

```
    --mp yes -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example in multiprocessing mode on all available CPUs by loading all data into memory and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols Source
  --mp yes --mpParams "inputDataMode,InMemory"
  -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example in multiprocessing mode on specific number of CPUs and chunk size without loading all data into memory and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols Source
  --mp yes --mpParams "inputDataMode,lazy,numProcesses,4,
  chunkSize,50" -i SampleChemspace.csv -o SampleChemspace.html
```

To run the first example using a set of specified parameters to generate fingerprints and LSH forest, configure faerun and scatter plot layout, and write out a HTML file containing TMAP visualization, type:

```
% VisualizeChemspaceUsingTMAP.py --categoricalDataCols Source
  --minHashFPParams "radius,3,numPermutations,2048"
  --lshForestParams "dim,2048,numPrefixTrees,128"
  --lshLayoutConfigParams "k,75,kc,20,slRepeats,2,
  slExtraScalingSteps,4,mmmRepeats,2"
  --faerunConfigParams "clearColor, #000000,thumbnailWidth, 250"
  --faerunScatterPlotParams "shader,circle,pointScale,4"
  --tmapDisplayMsg "TMAP Chemspace visualization"
  -i SampleChemspace.csv -o SampleChemspace.html
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

RDKitConvertFileFormat.py, RDKitCalculateMolecularDescriptors.py, RDKitStandardizeMolecules.py

COPYRIGHT

Copyright (C) 2026 Manish Sud. All rights reserved.

The functionality available in this script is implemented using TMAP and Faerun, open source software packages for visualizing chemspace, and RDKit, an open source toolkit for cheminformatics developed by Greg Landrum.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.