NAME

RDKitPerformTorsionScan.py - Perform torsion scan

SYNOPSIS

RDKitPerformTorsionScan.py [--addHydrogens <yes or no>] [--conformerGenerator <SDG, ETDG, KDG, ETKDG>] [--forceField <UFF, or MMFF>] [--forceFieldMMFFVariant <MMFF94 or MMFF94s>] [--enforceChirality <yes or no>] [--infile3D <yes or no>] [--infileParams <Name,Value,...>] [--modeMols <First or All>] [--modeTorsions <First or All>] [--maxConfs <number>] [--maxConfsTorsion <number>] [--maxIters <number>] [--mp <yes or no>] [--mpParams <Name.Value,...>] [--outfileMolName <yes or no>] [--outfileParams <Name,Value,...>] [--outPlotParams <Name,Value,...>] [--outPlotTitleTorsionSpec <yes or no>] [--overwrite] [--quiet <yes or no>] [--removeHydrogens <yes or no>] [--randomSeed <number>] [--torsionMaxMatches <number>] [--torsionMinimize <yes or no>] [--torsionRange <Start,Stop,Step>] [--useChirality <yes or no>] [--useTethers <yes or no>] [-w <dir>] -t <torsions> -i <infile> -o <outfile>

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

DESCRIPTION

Perform torsion scan for molecules around torsion angles specified using SMILES/SMARTS patterns. A molecule is optionally minimized before performing a torsion scan. A set of initial 3D structures are generated for a molecule by scanning the torsion angle across the specified range and updating the 3D coordinates of the molecule. A conformation ensemble is optionally generated for each 3D structure representing a specific torsion angle. The conformation with the lowest energy is selected to represent the torsion angle. An option is available to skip the generation of the conformation ensemble and simply calculate the energy for the initial 3D structure for a specific torsion angle

The torsions are specified using SMILES or SMARTS patterns. A substructure match is performed to select torsion atoms in a molecule. The SMILES pattern match must correspond to four torsion atoms. The SMARTS patterns containing atom indices may match more than four atoms. The atoms indices, however, must match exactly four torsion atoms. For example: [s:1][c:2]([aX2,cH1])!@[CX3:3](O)=[O:4] for thiophene esters and carboxylates as specified in Torsion Library (TorLib) [Ref 146].

A set of four output files is generated for each torsion match in each molecule. The names of the output files are generated using the root of the specified output file. They may either contain sequential molecule numbers or molecule names as shown below:

```
<OutfileRoot>_Mol<Num>.sdf
<OutfileRoot>_Mol<Num>_Torsion<Num>_Match<Num>.sdf
<OutfileRoot>_Mol<Num>_Torsion<Num>_Match<Num>_Energies.csv
<OutfileRoot>_Mol<Num>_Torsion<Num>_Match<Num>_Plot.<ImgExt>

or

<OutfileRoot>_<MolName>.sdf
<OutfileRoot>_<MolName>_Torsion<Num>_Match<Num>.sdf
<OutfileRoot>_<MolName>_Torsion<Num>_Match<Num>_Energies.csv
<OutfileRoot>_<MolName>_Torsion<Num>_Match<Num>_Plot.<ImgExt>
```

The supported input file formats are: Mol (.mol), SD (.sdf, .sd), .csv, .tsv .txt)

The supported output file formats are: SD (.sdf, .sd)

OPTIONS

-a, --addHydrogens <yes or no> [default: yes]

Add hydrogens before minimization.

-c, --conformerGenerator <SDG, ETDG, KDG, ETKDG> [default: ETKDG]

Conformation generation methodology for generating initial 3D structure of a molecule and conformation ensemble representing a specific torsion angle. No conformation ensemble is generated for 'No' value of '--torsionMinimize' option.

Possible values: Standard Distance Geometry, (SDG), Experimental Torsion-angle preference with Distance Geometry (ETDG), basic Knowledge-terms with Distance Geometry (KDG), and Experimental Torsion-angle preference along with basic Knowledge-terms with Distance Geometry (ETKDG) [Ref 129] .

-f, --forceField <UFF, MMFF> [default: MMFF]

Forcefield method to use for energy minimization of initial 3D structure of a molecule and conformation ensemble representing a specific torsion. No conformation ensemble is generated during for 'No' value of '--torsionMinimze' option and constrained energy minimization is not performed. Possible values: Universal Force Field (UFF) [Ref 81] or Merck Molecular Mechanics Force Field [Ref 83-87] .

--forceFieldMMFFVariant <MMFF94 or MMFF94s> [default: MMFF94]

Variant of MMFF forcefield to use for energy minimization.

--enforceChirality <yes or no> [default: Yes]

Enforce chirality for defined chiral centers during generation of conformers.

-e, --examples

Print examples.

-h, --help

Print this help message.

-i, --infile <infile>

Input file name.

--infile3D <yes or no> [default: no]

Skip generation and minimization of initial 3D structures for molecules in input file containing 3D coordinates.

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

A comma delimited list of parameter name and value pairs for reading molecules from files. The supported parameter names for different file formats, along with their default values, are shown below:

```
SD, MOL: removeHydrogens,yes,sanitize,yes,strictParsing,yes

SMILES: smilesColumn,1,smilesNameColumn,2,smilesDelimiter,space,
    smilesTitleLine,auto,sanitize,yes
```

Possible values for smilesDelimiter: space, comma or tab.

--modeMols <First or All> [default: First]

Perform torsion scan for the first molecule or all molecules in input file.

--modeTorsions <First or All> [default: First]

Perform torsion scan for the first or all specified torsion pattern in molecules up to a maximum number of matches for each torsion specification as indicated by '--torsionMaxMatches' option.

--maxConfs <number> [default: 250]

Maximum number of conformations to generate for initial 3D structure of a molecule. The lowest energy conformation is written to the output file.

--maxConfsTorsion <number> [default: 50]

Maximum number of conformations to generate for conformation ensemble representing a specific torsion. A constrained minimization is performed using the coordinates of the specified torsion and the lowest energy conformation is written to the output file.

--maxIters <number> [default: 500]

Maximum number of iterations to perform for a molecule during minimization to generation initial 3D structures. This option is ignored during 'yes' value of '--infile3D' option.

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

Use multiprocessing.

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

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

A comma delimited list of parameter name and value pairs for to configure multiprocessing.

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.

-o, --outfile <outfile>

Output file name. The output file root is used for generating the names of the output files corresponding to structures, energies, and plots during the torsion scan.

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

Append molecule name to output file root during the generation of the names for output files. The default is to use <MolNum>. The non alphabetical characters in molecule names are replaced by underscores.

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

A comma delimited list of parameter name and value pairs for writing molecules to files. The supported parameter names for different file formats, along with their default values, are shown below:

```
SD: kekulize,no
```

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

A comma delimited list of parameter name and value pairs for generating plots using Seaborn module. The supported parameter names along with their default values are shown below:

```
type,linepoint,outExt,svg,width,10,height,5.6,
title,auto,xlabel,auto,ylabel,auto,titleWeight,bold,labelWeight,bold
style,darkgrid,palette,deep,font,sans-serif,fontScale,1,
context,notebook
```

Possible values:

```
type: linepoint, scatter, or line. Both points and lines are drawn
    for linepoint plot type.
outExt: Any valid format supported by Python module Matplotlib.
    For example: PDF (.pdf), PNG (.png), PS (.ps), SVG (.svg)
titleWeight, labelWeight: Font weight for title and axes labels.
    Any valid value.
```

```
            style: darkgrid, whitegrid, dark, white, ticks
            palette: deep, muted, pastel, dark, bright, colorblind
            font: Any valid font name

         --outPlotTitleTorsionSpec <yes or no>  [default: yes]
       Append torsion specification to the title of the torsion plot.
```

--overwrite

Overwrite existing files.

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

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

--randomSeed <number> [default: auto]

Seed for the random number generator for generating initial 3D coordinates. Default is to use a random seed.

--removeHydrogens <yes or no> [default: Yes]

Remove hydrogens after minimization.

-t, --torsions <SMILES/SMARTS,...,...>

SMILES/SMARTS patterns corresponding to torsion specifications. It's a comma delimited list of valid SMILES/SMART patterns.

A substructure match is performed to select torsion atoms in a molecule. The SMILES pattern match must correspond to four torsion atoms. The SMARTS patterns contain atom indices may match more than four atoms. The atoms indices, however, must match exactly four torsion atoms. For example: [s:1][c:2]([aX2,cH1])!@[CX3:3](O)=[O:4] for thiophene esters and carboxylates as specified in Torsion Library (TorLib) [Ref 146].

--torsionMaxMatches <number> [default: 5]

Maximum number of torsions to match for each torsion specification in a molecule.

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

Perform constrained energy minimization on a conformation ensemble for a specific torsion angle and select the lowest energy conformation representing the torsion angle.

--torsionRange <Start,Stop,Step> [default: 0,360,5]

Start, stop, and step size angles in degrees for a torsion scan. In addition, you may specify values using start and stop angles from -180 to 180.

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

Use chirrality during substructure matches for identification of torsions. --useTethers <yes or no> [default: yes] Use tethers to optimize the final conformation by applying a series of extra forces to align matching atoms to the positions of the core atoms. Otherwise, use simple distance constraints during the optimization.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To perform a torsion scan on first molecule in a SMILES file using a minimum energy structure of the molecule selected from an ensemble of conformations, skipping generation of conformation ensembles for specific torsion angles and constrained energy minimization of the ensemble, generate output files corresponding to structure, energy and torsion plot, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC" -i SampleSeriesD3R.smi
  -o SampleOut.sdf
```

To run the previous example on all molecules in a SD file, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC" --modeMols All
  -i SampleSeriesD3R.sdf -o SampleOut.sdf
```

To perform a torsion scan on first molecule in a SMILES file using a minimum energy structure of the molecule selected from an ensemble of conformations, generation of conformation ensembles for specific torsion angles

and constrained energy minimization of the ensemble, generate output files corresponding to structure, energy and torsion plot, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC" --torsionMinimize Yes
   -i SampleSeriesD3R.smi -o SampleOut.sdf
```

To run the previous example on all molecules in a SD file, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC" --modeMols All
   --torsionMinimize Yes -i SampleSeriesD3R.sdf -o SampleOut.sdf
```

To run the previous example in multiprocessing mode on all available CPUs without loading all data into memory and write out a SD file, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC" -i SampleSeriesD3R.smi
  -o SampleOut.sdf --modeMols All --torsionMinimize Yes --mp yes
```

To run the previous example in multiprocessing mode on all available CPUs by loading all data into memory and write out a SD file, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC" -i SampleSeriesD3R.smi
  -o SampleOut.sdf --modeMols All --torsionMinimize Yes --mp yes
  --mpParams "inputDataMode,InMemory"
```

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

```
% RDKitPerformTorsionScan.py  -t "O=CNC" -i SampleSeriesD3R.smi
  -o SampleOut.sdf --modeMols All --torsionMinimize Yes --mp yes
  --mpParams "inputDataMode,Lazy,numProcesses,4,chunkSize,8"
```

To perform a torsion scan on first molecule in a SD file containing 3D coordinates, skipping generation of conformation ensembles for specific torsion angles and constrained energy minimization of the ensemble, generate output files corresponding to structure, energy and torsion plot, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC"  --infile3D yes
  -i SampleSeriesD3R3D.sdf -o SampleOut.sdf
```

To perform a torsion scan using multiple torsion specifications on all molecules in a SD file containing 3D coordinates, generation of conformation ensembles for specific torsion angles and constrained energy minimization of the ensemble, generate output files corresponding to structure, energy and torsion plot, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC,[O:1]=[C:2](c)[N:3][C:4]"
  --infile3D yes --modeMols All  --modeTorsions All
  --torsionMinimize Yes -i SampleSeriesD3R3D.sdf -o SampleOut.sdf
```

To run the previous example using a specific torsion scan range, type:

```
% RDKitPerformTorsionScan.py  -t "O=CNC,[O:1]=[C:2](c)[N:3][C:4]"
  --infile3D yes --modeMols All --modeTorsions All --torsionMinimize
  Yes --torsionRange 0,360,10 -i SampleSeriesD3R.smi -o SampleOut.sdf
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

RDKitCalculateRMSD.py, RDKitCalculateMolecularDescriptors.py, RDKitCompareMoleculeShapes.py, RDKitConvertFileFormat.py, RDKitPerformConstrainedMinimization.py

COPYRIGHT

The functionality available in this script is implemented using RDKit, an open source toolkit for cheminformatics developed by Greg Landrum.

This file is part of MayaChemTools.