NAME

RDKitStandardizeMolecules.py - Standardize molecules

SYNOPSIS

RDKitStandardizeMolecules.py [--infileParams <Name,Value,...>] [--methodologyParams <Name,Value,...>] [--mode <standardize or count>] [--mp <yes or no>] [--mpParams <Name,Value,...>] [--outfileParams <Name,Value,...>] [--overwrite] [--standardizeParams <Name,Value,...>] [--quiet <yes or no>] [-w <dir>] [-o <outfile>] -i <infile>

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

DESCRIPTION

Standardize molecules and write them out to an output file or simply count the number of molecules to be standardized. The standardization methodology consists of the following 4 steps executed in a sequential manner:

```
    1. Cleanup molecules
    2. Keep largest fragment
    3. Neutralize molecules
    4. Select canonical tautomer
```

The molecules are cleaned up by performing the following actions:

```
    1. Remove hydrogens
    2. Disconnect metal atoms - Disconnect metal atoms covalently bonded
       to non-metals
    3. Normalize - Normalize functional groups and recombine charges
    4. Reionize - Ionize strongest acid groups first in partially
       ionized molecules
    5. Assign stereochemistry
```

You may optionally skip any cleanup action during standardization.

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

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

OPTIONS

-e, --examples

Print examples.

-h, --help

Print this help message.

-i, --infile <infile>

Input file name.

--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,no,sanitize,yes,strictParsing,yes
    SMILES: smilesColumn,1,smilesNameColumn,2,smilesDelimiter,space,
        smilesTitleLine,auto,sanitize,yes
```

Possible values for smilesDelimiter: space, comma or tab.

-m, --mode <standardize or count> [default: standardize]

Specify whether to standardize molecules and write them out or simply count the number of molecules being standardized.

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

A comma delimited list of parameter name and value pairs to control the execution of different steps in the standardization methodology. The supported parameter names along with their default values are shown below:

```
    cleanup,yes,removeFragments,yes,neutralize,yes,
canonicalizeTautomer,yes
```

The standardization methodology consists of the following 4 steps executed in a sequential manner starting from step 1:

```
1. cleanup
2. removeFragments
3. neutralize
4. canonicalizeTautomer
```

You may optionally skip the execution of any standardization step.

The step1, cleanup, performs the following actions:

```
1. Remove hydrogens
2. Disconnect metal atoms - Disconnect metal atoms covalently bonded
    to non-metals
3. Normalize - Normalize functional groups and recombine charges
4. Reionize - Ionize strongest acid groups first in partially
    ionized molecules
5. Assign stereochemistry
```

You may optionally skip any cleanup action using '-s, --standardize' option.

The step2, removeFragments, employs rdMolStandardize.FragmentParent() function to keep the largest fragment.

The step3, neutralize, uses rdMolStandardize.Uncharger().uncharge() function to neutralize molecules by adding/removing hydrogens.

The step4, canonicalizeTautomer, relies on Canonicalize() function availabe via rdMolStandardize.TautomerEnumerator() to select a canonical tautomer.

--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 'chunkSize'.

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

A comma delimited list of parameter name and value pairs 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.

--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: compute2DCoords,auto,kekulize,yes
SMILES: smilesKekulize,no,smilesDelimiter,space, smilesIsomeric,yes,
    smilesTitleLine,yes,smilesMolName,yes,smilesMolProps,no
```

Default value for compute2DCoords: yes for SMILES input file; no for all other file types.

--overwrite

Overwrite existing files.

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

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

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

A comma delimited list of parameter name and value pairs for standardizing molecules. The supported parameter names along with their default values are shown below:

```
acidbaseFile,none,fragmentFile,none,normalizationsFile,none,
tautomerTransformsFile,none,
cleanupRemoveHydrogens,yes,cleanupDisconnectMetals,yes,
cleanupNormalize,yes,cleanupNormalizeMaxRestarts,200,
cleanupReionize,yes,cleanupAssignStereo,yes,
cleanupAssignStereoCleanIt,yes,cleanupAssignStereoForce,yes
largestFragmentChooserUseAtomCount,yes,
largestFragmentChooserCountHeavyAtomsOnly,no,preferOrganic,no,
doCanonical,yes,
maxTautomers,1000,maxTransforms,1000,
tautomerRemoveBondStereo,yes,tautomerRemoveIsotopicHs,yes
tautomerRemoveSp3Stereo,yes,tautomerReassignStereo,yes
```

A brief description of the standardization parameters, taken from RDKit documentation, is as follows:

```
acidbaseFile - File containing acid and base definitions
fragmentFile - File containing fragment definitions
normalizationsFile - File conataining normalization transformations
tautomerTransformsFile - File containing tautomer transformations

cleanupRemoveHydrogens - Remove hydrogens druring cleanup
cleanupDisconnectMetals - Disconnect metal atoms covalently bonded
    to non-metals during cleanup
cleanupNormalize - Normalize functional groups and recombine
    charges during cleanup
cleanupNormalizeMaxRestarts - Maximum number of restarts during
    normalization step of cleanup
cleanupReionize -Ionize strongest acid groups first in partially
    ionized molecules during cleanup
cleanupAssignStereo - Assign sterochemistry during cleanup
cleanupAssignStereoCleanIt - Clean property _CIPCode during
    assign stereochemistry
cleanupAssignStereoForce - Always perform stereochemistry
    calculation during assign stereochemistry

largestFragmentChooserUseAtomCount - Use atom count as main
```

```
        criterion before molecular weight to determine largest fragment
        in LargestFragmentChooser
    largestFragmentChooserCountHeavyAtomsOnly - Count only heavy
        atoms to determine largest fragment in LargestFragmentChooser
    preferOrganic - Prefer organic fragments over  inorganic ones when
        choosing fragments

    doCanonical - Apply atom-order dependent normalizations in a
        canonical order during uncharging

    maxTautomers - Maximum number of tautomers to generate
    maxTransforms - Maximum number of transforms to apply during
        tautomer enumeration
    tautomerRemoveBondStereo - Remove stereochemistry from double bonds
        involved in tautomerism
    tautomerRemoveIsotopicHs: Remove isotopic Hs from centers involved in tautomerism
    tautomerRemoveSp3Stereo - Remove stereochemistry from sp3 centers
        involved in tautomerism
    tautomerReassignStereo - AssignStereochemistry on all generated tautomers
```

The default value is set to none for the following file name parameters: acidbaseFile, fragmentFile, normalizationsFile, and tautomerTransformsFile. The script relies on RDKit to automatically load appropriate acid base and fragment definitions along with normalization and tautomer transformations from a set of internal catalogs.

Note: The fragmentFile doesn't appear to be used by the RDKit method rdMolStandardize.FragmentParent() to find largest fragment.

The contents of various standardization definitions and transformations files are described below:

```
    acidbaseFile - File containing acid and base definitions

        // Name     Acid                 Base
        -OSO3H      OS(=O)(=O)[OH]       OS(=O)(=O)[O-]
        -SO3H       [!O]S(=O)(=O)[OH]    [!O]S(=O)(=O)[O-]
        -OSO2H      O[SD3](=O)[OH]       O[SD3](=O)[O-]
        ... ... ...

    fragmentFile - File containing fragment definitions

        // Name     SMARTS
        hydrogen     [H]
        fluorine     [F]
        chlorine     [Cl]
        ... ... ...

    normalizationsFile - File conataining normalization transformations

        // Name     SMIRKS
        Sulfone to S(=O)(=O)        [S+2:1]([O-:2])([O-:3])>>
            [S+0:1](=[O-0:2])(=[O-0:3])
        Pyridine oxide to n+O-     [n:1]=[O:2]>>[n+:1][O-:2]
        ... ... ...

    tautomerTransformsFile - File containing tautomer transformations

        // Name                 SMARTS   Bonds  Charges
        1,3 (thio)keto/enol f  [CX4!H0]-[C]=[O,S,Se,Te;X1]
        1,3 (thio)keto/enol r  [O,S,Se,Te;X2!H0]-[C]=[C]
        1,5 (thio)keto/enol f  [CX4,NX3;!H0]-[C]=[C][CH0]=[O,S,Se,Te;X1]
        ... ... ...
```

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To standardize molecules in a SMILES file by executing all standardization steps and write out a SMILES file, type:

```
    % RDKitStandardizeMolecules.py -i Sample.smi -o SampleOut.smi
```

To standardize molecules in a SD file by executing all standardization steps, performing standardization in multiprocessing mode on all available CPUs without loading all data into memory, and write out and write out a SD file, type:

```
    % RDKitStandardizeMolecules.py --mp yes -i Sample.sdf -o SampleOut.sdf
```

To standardize molecules in a SMILES file by executing all standardization steps, performing standardization in multiprocessing mode on all available CPUs by loading all data into memory, and write out and write out a SMILES file, type:

```
    % RDKitStandardizeMolecules.py --mp yes --mpParams "inputDataMode,
      InMemory" -i Sample.smi -o SampleOut.smi
```

To standardize molecules in a SMILES file by executing all standardization steps, performing standardization in multiprocessing mode on specific number of CPUs and chunk size without loading all data into memory, and write out a a SMILES file, type:

```
    % RDKitStandardizeMolecules.py --mp yes --mpParams "inputDataMode,Lazy,
      numProcesses,4,chunkSize,8" -i Sample.smi -o SampleOut.smi
```

To count number of molecules to be standardized without generating any output file, type:

```
    % RDKitStandardizeMolecules.py -m count -i Sample.sdf
```

To standardize molecules in a SD file by executing specific standardization steps along with explicit values for various parameters to control the standardization behavior, and write out a SD file, type:

```
    % RDKitStandardizeMolecules.py --methodologyParams "cleanup,yes,
      removeFragments,yes,neutralize,yes,canonicalizeTautomer,yes"
      --standardizeParams "cleanupRemoveHydrogens,yes,
      cleanupDisconnectMetals,yes,cleanupNormalize,yes,
      cleanupNormalizeMaxRestarts,200,cleanupReionize,yes,
      cleanupAssignStereo,yes,largestFragmentChooserUseAtomCount,yes,
      doCanonical,yes,maxTautomers,1000"
      -i Sample.sdf -o SampleOut.sdf
```

To standardize molecules in a CSV SMILES file, SMILES strings in column 1, name in column 2, and generate output SD file, type:

```
    % RDKitStandardizeMolecules.py --infileParams
      "smilesDelimiter,comma,smilesTitleLine,yes,smilesColumn,1,
      smilesNameColumn,2" --outfileParams "compute2DCoords,yes"
      -i SampleSMILES.csv -o SampleOut.sdf
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

RDKitConvertFileFormat.py, RDKitEnumerateTautomers.py, RDKitRemoveDuplicateMolecules.py, RDKitRemoveInvalidMolecules.py, RDKitRemoveSalts.py, RDKitSearchFunctionalGroups.py, RDKitSearchSMARTS.py

COPYRIGHT