NAME

RDKitPerformRGroupDecomposition.py - Perform R group decomposition analysis

SYNOPSIS

RDKitPerformRGroupDecomposition.py [--coreScaffold <ByMCS, BySMARTS or BySMILES>] [--decompositionParams <Name,Value,...>] [--infileParams <Name,Value,...>] [--mcsParams <Name,Value,...>] [--outfileParams <Name,Value,...>] [--overwrite] [--quote <yes or no>] [--removeUnmatched <yes or no>] [--smartsOrSmilesCoreScaffold <text>] [-w <dir>] -i <infile> -o <outfile>

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

DESCRIPTION

Perform R group decomposition for a set of molecules in a series containing a common core scaffold. The core scaffold is identified by a SMARTS string, SMILES string, or using maximum common substructure (MCS) search. Multiple core scaffolds may be specified using SMARTS or SMILES strings for set of molecules corresponding to multiple series.

The core scaffolds along with appropriate R groups are written out as SMILES strings to a SD or text file. The unmatched molecules without any specified core scaffold are written to a different output file.

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

The supported output file formats are: SD (.sdf, .sd), CSV/TSV (.csv, .tsv, .txt)

OPTIONS

-c, --coreScaffold <ByMCS, BySMARTS or BySMILES> [default: ByMCS]

Specify a core scaffold for a set of molecules in a series. The core scaffold is identified by an explicit SMARTS string, SMILES string, or using maximum common substructure (MCS) search. Multiple core scaffolds may be specified using SMARTS or SMILES strings for set of molecules corresponding to multiple series.

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

Parameter values to use during R group decomposition for a series of molecules. In general, it is a comma delimited list of parameter name and value pairs. The supported parameter names along with their default values are shown below:

```
RGroupCoreAlignment,MCS, RGroupMatching,GreedyChunks,chunkSize,5,
matchOnlyAtRGroups,no,removeHydrogenOnlyGroups,yes,
removeHydrogensPostMatch,no
```

A brief description of each supported parameter taken from RDKit documentation, along with their possible values, is as follows.

RGroupCoreAlignment - Mapping of core labels:

```
MCS - Map core labels to each other using MCS
None - No mapping
```

RGroupMatching: Greedy, GreedyChunks, Exhaustive

matchOnlyAtRGroups - Allow R group decomposition only at specified R groups. Possible values: yes, no.

removeHydrogenOnlyGroups - Remove all R groups that only have hydrogens. Possible values: yes, no.

removeHydrogensPostMatch - Remove all hydrogens from the output molecules. Possible values: yes, no.

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

Possible values for smilesDelimiter: space, comma or tab.

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

Parameter values to use for identifying a maximum common substructure (MCS) in a series of molecules. In general, it is a comma delimited list of parameter name and value pairs. The supported parameter names along with their default values are shown below:

```
atomCompare,CompareElements,bondCompare,CompareOrder,
maximizeBonds,yes,matchValences,yes,matchChiralTag,no,
minNumAtoms,1,minNumBonds,0,ringMatchesRingOnly,yes,
completeRingsOnly,yes,threshold,1.0,timeOut,3600,seedSMARTS,none
```

Possible values for atomCompare: CompareAny, CompareElements, CompareIsotopes. Possible values for bondCompare: CompareAny, CompareOrder, CompareOrderExact.

A brief description of MCS parameters taken from RDKit documentation is as follows:

```
atomCompare - Controls match between two atoms
bondCompare - Controls match between two bonds
maximizeBonds - Maximize number of bonds instead of atoms
matchValences - Include atom valences in the MCS match
matchChiralTag - Include atom chirality in the MCS match
minNumAtoms - Minimum number of atoms in the MCS match
minNumBonds - Minimum number of bonds in the MCS match
ringMatchesRingOnly - Ring bonds only match other ring bonds
completeRingsOnly - Partial rings not allowed during the match
threshold - Fraction of the dataset that must contain the MCS
seedSMARTS - SMARTS string as the seed of the MCS
timeout - Timeout for the MCS calculation in seconds
```

-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,forceV3000,no
SMILES: smilesKekulize,no,smilesIsomeric,yes
```

Default value for compute2DCoords: yes for SMILES input file; no for all other file types. The kekulize and smilesIsomeric parameters are also used during generation of SMILES strings for CSV/TSV files.

--overwrite

Overwrite existing files.

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

Quote SMILES strings and molecule names before writing them out to text files. Possible values: yes or no. Default: yes for CSV (.csv) text files; no for TSV (.tsv) and TXT (.txt) text files.

-r, --removeUnmatched <yes or no> [default: no]

Remove unmatched molecules containing no specified core scaffold from the output file and write them to a different output file.

-s, --smartsOrSmilesCoreScaffold <text> [default: none]

SMARTS or SMILES string to use for core scaffold during 'SMARTS' or 'SMILES' value of '-c, --coreScaffold' option. Multiple core scaffolds may be specified using a comma delimited set of SMARTS or SMILES strings.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To perform R group decomposition for a set of molecules in a series using MCS to identify a core scaffold and write out a CSV file containing R groups, type:

```
% RDKitPerformRGroupDecomposition.py -i SampleSeriesD3R.smi
  -o SampleSeriesD3ROut.csv
```

To perform R group decomposition for a set of molecules in a series using a specified core scaffold and write out a SD file containing R groups, type:

```
% RDKitPerformRGroupDecomposition.py  -c BySMARTS
  -s "Nc1nccc(-c2cnc(CNCc3ccccc3)c2)n1" -i SampleSeriesD3R.smi
  -o SampleSeriesD3ROut.sdf
```

To perform R group decomposition for a set of molecules in a series using MCS to identify a core scaffold and write out CSV files containing matched and unmatched molecules without quoting values, type:

```
% RDKitPerformRGroupDecomposition.py -c ByMCS -r yes -q no
  -i SampleSeriesD3R.sdf -o SampleSeriesD3ROut.csv
```

To perform R group decomposition for a set of molecules in multiple series using specified core scaffolds and write out a TSV file containing R groups, type:

```
% RDKitPerformRGroupDecomposition.py  -c BySMARTS
  -s "Nc1nccc(-c2cnc(CNCc3ccccc3)c2)n1,[#6]-[#6]1:[#6]:[#6]:[#6]:[#6]:
  [#6]:1" -i SampleMultipleSeriesD3R.smi -o
  SampleMultipleSeriesD3ROut.tsv
```

To perform R group decomposition for a set of molecules in a CSV SMILES file, SMILES strings in olumn 1, name in column 2, and write out a CSV file containing R groups, type:

```
% RDKitPerformRGroupDecomposition.py --infileParams
  "smilesDelimiter,comma,smilesTitleLine,yes,smilesColumn,1,
  smilesNameColumn,2" --outfileParams "compute2DCoords,yes"
  -i SampleSeriesD3R.smi -o SampleSeriesD3ROut.csv
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

RDKitConvertFileFormat.py, RDKitSearchFunctionalGroups.py, RDKitSearchSMARTS.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.

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.