NAME

RDKitPerformPositionalAnalogueScan.py - Positional analogue scanning.

SYNOPSIS

RDKitPerformPositionalAnalogueScan.py [--comboSearchAtoms <number>] [--clearAtomMapNumbers <yes or no>] [--infileParams <Name,Value,...>] [--mode <ReplaceAtoms, AttachAtoms, AttachSMILES, RxnSMARTS, TandemScan>] [--mp <yes or no>] [--mpParams <Name,Value,...>] [--nameSuffix <text>] [--outfileParams <Name,Value,...>] [--overwrite] [--permuteTandemScan <yes or no>] [--quiet <yes or no>] [--searchPattern <SMARTSPattern>] [--targetPattern <ElementSymbol, SMILES, RxnSMARTS,...>] [--uniqueAnalogues <yes or no>] [-w <dir>] -i <infile> -o <outfile>

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

DESCRIPTION

Perform Positional Analogue Scanning (PAS) to generate analogues of molecules by applying chemical transformations to molecules [Ref 147-148]. The chemical transformations are defined using SMARTS, element symbols, SMILES, and RxnSMARTS. Four different types of chemical transformations are available for for generating analogues of molecules: replace atoms, attach atoms, attach SMILES, and RxnSMARTS. Tandem positional analogue scanning may be performed by the concurrent application of multiple chemical transformations.

A SMARTS search pattern identifies atoms in molecules for attachment or replacement points during positional analogue scanning. It may retrieve multiple substructure matches in a molecule. The first matched atom in each substructure match comprises a set of attachment or replacement points.

A target pattern encompasses information regarding element symbol, SMILES, and reaction SMARTS for replacing and attaching atoms, attaching SMILES, and applying reaction SMARTS. In addition, multiple concurrent chemical transformations may be specified during tandem positional analogue scanning.

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

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

OPTIONS

-c, --comboSearchAtoms <number> [default: 1]

Number of concurrent search pattern match atoms to use as attachment or replacement points during positional analogue scanning in 'AttachAtoms', 'AttachSMILES', and 'ReplaceAtoms' modes. This value is ignored during 'RxnSMARTS' and 'TandemScan' values of '-m, --mode' option.

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

Clear any atom map numbers in target SMILES specification for 'AttachSMILES' before generating analogues.

-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, --mode <ReplaceAtoms, AttachAtoms,...> [default: ReplaceAtoms]

Type of operations to perform during positional analogue scanning. The supported values, along with a brief description, are shown below:

```
Mode           Description
ReplaceAtoms   Replace atoms with new atoms
AttachAtoms    Attach new atoms to atoms
AttachSMILES   Attach SMILES to atoms
RxnSMARTS      Run reaction SMARTS
TandemScan     Perform tandem scan by combining ReplaceAtoms,
    AttachAtoms, and AttachSMILES
```

The chemical transformations of input molecules is dependent on the values of '-s, --searchPattern' and '-t, --targetPattern' options. For example, nitrogen-walk or nitrogen scan is performed by '[cH]' and 'N' values for '-s, --searchPattern' and '-t, --targetPattern' options.

--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.

-n, --nameSuffix <text> [default: Analogue]

Name suffix for generating molecule names of analogues. Format of analogue names:

-o, --outfile <MolName>_<NameSuffix>_<MolNum>

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,smilesDelimiter,space, smilesIsomeric,yes,
    smilesTitleLine,yes,smilesMolName,yes,smilesMolProps,no
```

--overwrite

Overwrite existing files.

-p, --permuteTandemScan <yes or no> [default: yes]

Permute atom positions matched by SMARTS search pattern in a molecule to generate all possible analogues during tandem positional analogue scanning.

This option is only valid for 'TandemScan' value of '-m, --mode' option.

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

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

-s, --searchPattern <SMARTSPattern> [default: auto]

SMARTS search pattern identifying atoms in molecules for attachment or replacement points during positional analogue scanning. The SMARTS search pattern may retrieve multiple substructure matches in a molecule. The first matched atom in each substructure match comprises a set of attachment or replacement points.

The default values, dependent on the value of '-m, --mode' option, are shown below:

```
Mode            Default   Description
ReplaceAtoms    [cH]      Aromatic carbon
AttachAtoms     [cH]      Aromatic carbon
AttachSMILES    [cH]      Aromatic carbon
RxnSMARTS       None      Not applicable
TandemScan      [cH]      Aromatic carbon
```

This option is ignored during 'RxnSMARTS' value of '-m, --mode' option.

-t, --targetPattern <ElementSymbol, SMILES, RxnSMARTS...> [default: auto]

Target pattern for performing chemical transformations during positional analogue scanning. These values are used in conjunction with the value of '-s, --searchPattern' to generate appropriate analogues.

The default values, dependent on the values of '-m, --mode' and '-s, --searchPattern' options, are shown below:

```
Mode            Default        Description
ReplaceAtoms    N              Element symbol for nitrogen
AttachAtoms     F              Element symbol for fluorine
AttachSMILES    C(F)(F)(F)     SMILES for CF3
RxnSMARTS       [cH:1]>>[N:1] Replace aromatic carbon by nitrogen
TandemScan      ReplaceAtoms,N,AttachAtoms,F  Replace and attach
```

The target specifications may contain multiple space delimted values. For example:

```
Mode            Example
ReplaceAtoms    "N P"
AttachAtoms     "F Cl Br"
AttachSMILES    "C(F)(F)(F) CO"
RxnSMARTS       "[cH:1]>>[N:1] [cH:1]>>[P:1]"
TandemScan      "ReplaceAtoms,N P,AttachAtoms,F Cl"
```

The target specification for 'AttachSMILES' may contain atom map numbers. The attachment point corresponds to the matched atom with the lowest atom map number. Otherwise, the first atom in SMILES target specification represents the attachment point. For example:

```
Mode            Example
AttachSMILES    "C(C)(C)(C) CO"
AttachSMILES    "C([C:1])(C)(C) C(F)(F)(F)"
```

Multiple concurrent chemical transformations are allowed during 'TandemScan'. The target pattern specification for 'TandemScan' is a comma delimited list of operation type and target pattern. Format: OperationType,TargetPattern,...

The supported operation types and target pattern for 'TandemScan' are shown below:

```
ReplaceAtoms,<ElementSymbol>
AttachAtoms,<ElementSymbol>
AttachSMILES,<SMILES>

ReplaceAtoms,"<ElementSymbol> <ElementSymbol>..."
AttachAtoms,"<ElementSymbol> <ElementSymbol>..."
AttachSMILES,"<SMILES> <SMIELS>..."
```

For example:

```
ReplaceAtoms,N,AttachAtoms,F
ReplaceAtoms,N,AttachAtoms,F,AttachSMILES,C(F)(F)(F)

"ReplaceAtoms,N,AttachAtoms,F Cl"
"ReplaceAtoms,N P,AttachAtoms,F Cl"
"ReplaceAtoms,N P,AttachAtoms,F Cl,AttachSMILES,C(F)(F)(F) CO"
```

The number of chemical transformations in 'TandemScan' must be less than or equal to the total number atoms matched by SMARTS search pattern in a molecule. Otherwise, it is not possible to perform a 'TandemScan'. The matched atom positions may be optionally permuted to generate all possible analogues during positional analogue scanning using '-p, --permuteTandemScan' option.

`-u, --uniqueAnalogues <yes or no> [default: yes]`

Keep only unique analogues of a molecule corresponding to unique SMILES strings. The duplicate SMILES string may be generated during PAS due to symmetric replacement or attachment points in molecules.

`-w, --workingdir <dir>`

Location of working directory which defaults to the current directory.

EXAMPLES

To perform a nitrogen-walk or nitrogen scan by replacing aromatic carbons by nitrogens in molecules in a SMILES file and write out a SMILES file, type:

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

To run the first example by explicitly specifying search and target patterns for replacing aromatic carbons by nitogens in molecules in a SD file and write out a SD file, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m ReplaceAtoms -s "[cH]"
  -t "N" -i Sample.sdf -o SampleOut.sdf
```

To run the previous example for replacing aromatic carbons by N and P, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m ReplaceAtoms -s "[cH]"
  -t "N P" -i Sample.sdf -o SampleOut.sdf
```

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

```
% RDKitPerformPositionalAnalogueScan.py  -i Sample.smi -o SampleOut.sdf
  --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:

```
% RDKitPerformPositionalAnalogueScan.py  -i Sample.smi -o SampleOut.sdf
  --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:

```
% RDKitPerformPositionalAnalogueScan.py  -i Sample.smi -o SampleOut.sdf
  --mpParams "inputDataMode,Lazy,numProcesses,4,chunkSize,8"
```

To perform positional analogue scanning by simultaneously attaching fluorines to two aromatic carbons in molecules in a SMILES file and write out a SD file, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m AttachAtoms -s "[cH]"
  -t "F" -c 2 -i Sample.smi -o SampleOut.sdf
```

To perform positional analogue scanning by attaching SMILES to aromatic carbons in molecules in a SMILES file and write out a SD file, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m AttachSMILES -s "[cH]"
  -t "F" -i Sample.smi -o SampleOut.sdf
```

To run the previous example for attaching F and CF3 to aromatic carbons, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m AttachSMILES -s "[cH]"
  -t "F C(F)(F)(F)" -i Sample.smi -o SampleOut.sdf
```

To run the previous example for attaching "C(C)(C)(C)" at a specific attachment point corresponding to the lowest atom map number, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m AttachSMILES -s "[cH]"
  -t "C([C:1])([C:2])(C)" -i Sample.smi -o SampleOut.sdf
```

To perform a nitrogen-walk or nitrogen scan by using reaction SMARTS to replace aromatic carbons by nitrogens in molecules in a SMILES file and write out a SMILES file, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m RxnSMARTS
  -t "[cH:1]>>[N:1]" -i Sample.smi -o SampleOut.smi
```

To perform a tandem positional analogue scan by concurrently applying multiple chemical transformations to aromatic carbons, permute all matched search atom positions during analogue generation, and write out a SD file, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m TandemScan -s "[cH]"
  -t "ReplaceAtoms,N,AttachAtoms,F,AttachSMILES,OC"
  -p yes  -i Sample.smi -o SampleOut.smi
```

To run the previous example for attaching multiple atoms and SMILES to aromatic carbons, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m TandemScan -s "[cH]"
  -t "ReplaceAtoms,N,AttachAtoms,F Cl,AttachSMILES,C(F)(F)(F) OC"
  -p yes  -i Sample.smi -o SampleOut.smi
```

To perform a nitrogen-walk or nitrogen scan by replacing aromatic carbons by nitrogens in molecules in a SMILES CSV fileS, MILES strings in column 1, name in column 2, and write out a SD file, type:

```
% RDKitPerformPositionalAnalogueScan.py  -m ReplaceAtoms -s "[cH]"
  -t "N" --infileParams "smilesDelimiter,comma, smilesTitleLine,yes,
  smilesColumn,1,smilesNameColumn,2"
  -i SampleSMILES.csv -o SampleOut.sdf
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

RDKitConvertFileFormat.py, RDKitEnumerateCompoundLibrary.py, RDKitPerformTorsionScan.py

COPYRIGHT

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

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.