NAME

RDKitCalculatePartialCharges.py - Calculate partial atomic charges

SYNOPSIS

RDKitCalculatePartialCharges.py [--allowParamFailure <yes or no>] [--chargesSDFormat <AtomAliases or DataField>] [--dataFieldLabel <text>] [--infileParams <Name,Value,...>] [--mode <Gasteiger or MMFF>] [--mp <yes or no>] [--mpParams <Name.Value,...>] [--numIters <number>] [--outfileParams <Name,Value,...>] [--precision <number>] [--overwrite] [-w <dir>] -i <infile> -o <outfile>

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

DESCRIPTION

Calculate partial charges for atoms in molecules and write them out to a SD file. The hydrogens are automatically added to molecules before calculating partial charges.

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

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

OPTIONS

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

Allow calculation of Gasteiger partial charges to proceed for molecules containing atoms with unknown parameters. The atoms with unknown parameters are removed from the calculations by setting their values to zero.

-c, --chargesSDFormat <AtomAliases or DataField> [default: AtomAliases]

Format for writing out partial atomic charges to SD file. Possible values: AtomAliases or DataField.

The charges are stored as atom property named 'molFileAlias' for 'AtomAliases' format and may be retrieved using the RDKit function 'GetProp' for atoms: Aotm.GetProp('molFileAliases').

The charges are stored under a data field label speciefied using '-d, --dataFieldLabel' for 'DataField' format and may be retrieved using the RDKit function 'GetProp' for molecues.

-d, --dataFieldLabel <text> [default: PartialCharges]

Data field label to use for storing charged in SD file during 'DataField' value of '-c, --chargesSDFormat'.

-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 <Gasteiger or MMFF> [default: Gasteiger]

Type of partial atomic charges to calculate. Possible values: Gasteiger [Ref 138] or Merk Molecular Mechanics Fore Field (MMFF) [Ref 83-87].

--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 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.

-n, --numIters <number> [default: 12]

Number of iterations to perform during calculation of Gasteiger charges.

-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,no
```

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

-p, --precision <number> [default: 3]

Floating point precision for writing the calculated partial atomic charges.

--overwrite

Overwrite existing files.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To calculate Gasteiger partial atomic charges for molecules in a SMILES file and write them out to a SD file as atom aliases, type:

```
% RDKitCalculatePartialCharges.py  -i Sample.smi -o SampleOut.sdf
```

To calculate Gasteiger partial atomic charges for molecules in a SMILES file in multiprocessing mode on all available CPUs without loading all data into memory, and and write them out to a SD file as atom aliases, type:

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

To calculate Gasteiger partial atomic charges for molecules in a SMILES file in multiprocessing mode on all available CPUs by loading all data into memory, and and write them out to a SD file as atom aliases, type:

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

To calculate Gasteiger partial atomic charges for molecules in a SMILES file in multiprocessing mode on specific number of CPUs without loading all data into memory, and and write them out to a SD file as atom aliases, type:

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

To calculate MMFF forcefield partial atomic charges for molecules in a SD file and write them out to a SD file under 'PartialCharges' data field, type:

```
% RDKitCalculatePartialCharges.py  -m MMFF -c DataField -i Sample.sdf
  -o SampleOut.sdf
```

To calculate Gasteiger partial atomic charges for molecules in a SMILES file and write them out to a SD file under a data field named 'GasteigerCharges', type:

```
% RDKitCalculatePartialCharges.py  -m Gasteiger -c DataField
  -d GasteigerCharges -p 4 -i Sample.smi -o SampleOut.sdf
```

To calculate Gasteiger partial atomic charges for molecules in a CSV SMILES file, SMILES strings in column 1, name in column 2, and write out a SD file containing charges as atom aliases, type:

```
% RDKitCalculatePartialCharges.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

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

COPYRIGHT

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