NAME

Psi4VisualizeDualDescriptors.py - Visualize dual descriptors for frontier orbitals

SYNOPSIS

Psi4VisualizeDualDescriptors.py [--basisSet <text>] [--infileParams <Name,Value,...>] [--methodName <text>] [--mode <GenerateCubeFiles, VisualizeCubeFiles, Both>] [--mp <yes or no>] [--mpParams <Name, Value,...>] [--outfilesDir <text>] [--outfilesMolPrefix <MolNum, MolName, Both>] [--overwrite] [--psi4CubeFilesParams <Name,Value,...>] [--psi4OptionsParams <Name,Value,...>] [--psi4RunParams <Name,Value,...>] [--pymolViewParams <Name,Value,...>] [--quiet <yes or no>] [--reference <text>] [-w <dir>] -i <infile> -o <outfile>

Psi4VisualizeDualDescriptors.py -h | --help | -e | --examples

DESCRIPTION

Generate and visualize dual descriptors [Ref 151] corresponding to frontier molecular orbitals for molecules in input file. A set of cube files, corresponding to dual descriptors, is generated for molecules. The cube files are used to create a PyMOL visualization file for viewing volumes, meshes, and surfaces representing dual descriptors for frontier molecular orbitals. An option is available to skip the generation of new cube files and use existing cube files to visualize frontier molecular orbitals.

The dual descriptor, a measure of electrophilicity and nucleophilicity for a molecule, is calculated by Psi4 from frontier molecular orbitals [Ref 151]: f2(r) = rhoLUMO(r) - rhoHOMO(r). The sign of the dual descriptor value corresponds to electrophilic and nucleophilic sites in molecules as shown below:

```
Sign          Site              Reactivity
Positive      Electrophilic     Favored for a nucleophilic attack
Negative      Nucleophilic      Favored for a electrophilic attack
```

A Psi4 XYZ format geometry string is automatically generated for each molecule in input file. It contains atom symbols and 3D coordinates for each atom in a molecule. In addition, the formal charge and spin multiplicity are present in the the geometry string. These values are either retrieved from molecule properties named 'FormalCharge' and 'SpinMultiplicty' or dynamically calculated for a molecule.

A set of cube and SD output files is generated for each closed-shell molecule in input file as shown below:

```
Ouput dir: <OutfileRoot>_DualDescriptors or <OutfilesDir>


<MolIDPrefix>.sdf
<MolIDPrefix>*DUAL*.cube
```

In addition, a <OutfileRoot>.pml is generated containing dual descriptors for frontier molecular orbitals for all molecules in input file.

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

The supported output file formats are: PyMOL script file (.pml)

A variety of PyMOL groups and objects are created for visualization of dual descriptors for closed-shell molecules as shown below:

```
<MoleculeID>
    .Molecule
        .Molecule
        .BallAndStick
    .Dual
        .Cube
        .Volume
        .Mesh
            .Positive_Electrophilic
            .Negative_Nucleophilic
        .Surface
            .Positive_Electrophilic
            .Negative_Nucleophilic
<MoleculeID>
    .Molecule
        ... ... ...
    .Dual
        ... ... ...
```

OPTIONS

-b, --basisSet <text> [default: auto]

Basis set to use for calculating single point energy before generating cube files corresponding to dual descriptors for frontier molecular orbitals. Default: 6-31+G** for sulfur containing molecules; Otherwise, 6-31G** [Ref 150]. The specified value must be a valid Psi4 basis set. No validation is performed.

The following list shows a representative sample of basis sets available in Psi4:

```
STO-3G, 6-31G, 6-31+G, 6-31++G, 6-31G*, 6-31+G*,  6-31++G*,
6-31G**, 6-31+G**, 6-31++G**, 6-311G, 6-311+G, 6-311++G,
6-311G*, 6-311+G*, 6-311++G*, 6-311G**, 6-311+G**, 6-311++G**,
cc-pVDZ, cc-pCVDZ, aug-cc-pVDZ, cc-pVDZ-DK, cc-pCVDZ-DK, def2-SVP,
def2-SVPD, def2-TZVP, def2-TZVPD, def2-TZVPP, def2-TZVPPD
```

-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
```

-m, --methodName <text> [default: auto]

Method to use for calculating single point energy before generating cube files corresponding to dual descriptors for frontier molecular orbitals. Default: B3LYP [Ref 150]. The specified value must be a valid Psi4 method name. No validation is performed.

The following list shows a representative sample of methods available in Psi4:

```
B1LYP, B2PLYP, B2PLYP-D3BJ, B2PLYP-D3MBJ, B3LYP, B3LYP-D3BJ,
B3LYP-D3MBJ, CAM-B3LYP, CAM-B3LYP-D3BJ, HF, HF-D3BJ,  HF3c, M05,
M06, M06-2x, M06-HF, M06-L, MN12-L, MN15, MN15-D3BJ,PBE, PBE0,
PBEH3c, PW6B95, PW6B95-D3BJ, WB97, WB97X, WB97X-D, WB97X-D3BJ
```

--mode <GenerateCubeFiles, VisualizeCubeFiles, or Both> [default: Both]

Generate and visualize cube files corresponding to dual descriptors for frontier molecular orbitals. The 'VisualizeCubes' value skips the generation of new cube files and uses existing cube files for visualization of dual descriptors. Multiprocessing is not supported during 'VisualizeCubeFiles' value of '--mode' 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 '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 for PyMOL PML file. The PML output file, along with cube files, is generated in a local directory corresponding to '--outfilesDir' option.

--outfilesDir <text> [default: auto]

Directory name containing PML and cube files. Default: <OutfileRoot>_DualDescriptors. This directory must be present during 'VisualizeCubeFiles' value of '--mode' option.

--outfilesMolPrefix <MolNum, MolName, Both> [default: Both]

Molecule prefix to use for the names of cube files. Possible values: MolNum, MolName, or Both. By default, both molecule number and name are used. The format of molecule prefix is as follows: MolNum - Mol<Num>; MolName - <MolName>, Both: Mol<Num>_<MolName>. Empty molecule names are ignored. Molecule numbers are used for empty molecule names.

--overwrite

Overwrite existing files.

--psi4CubeFilesParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for generating Psi4 cube files.

The supported parameter names along with their default and possible values are shown below:

```
gridSpacing, 0.2, gridOverage, 4.0, isoContourThreshold, 0.85
```

gridSpacing: Grid spacing for generating cube files. Units: Bohr. A higher value reduces the size of the cube files on the disk. This option corresponds to Psi4 option CUBIC_GRID_SPACING.

gridOverage: Grid overage for generating cube files. Units: Bohr.This option corresponds to Psi4 option CUBIC_GRID_OVERAGE.

isoContourThreshold: IsoContour values for generating cube files that capture specified percent of the probability density using the least amount of grid points. Default: 0.85 (85%). This option corresponds to Psi4 option CUBEPROP_ISOCONTOUR_THRESHOLD.

--psi4OptionsParams <Name,Value,...> [default: none]

A comma delimited list of Psi4 option name and value pairs for setting global and module options. The names are 'option_name' for global options and 'module_name__option_name' for options local to a module. The specified option names must be valid Psi4 names. No validation is performed.

The specified option name and value pairs are processed and passed to psi4.set_options() as a

dictionary. The supported value types are float, integer, boolean, or string. The float value string is converted into a float. The valid values for a boolean string are yes, no, true, false, on, or off.

--psi4RunParams <Name,Value,...> [default: auto]

A comma delimited list of parameter name and value pairs for configuring Psi4 jobs.

The supported parameter names along with their default and possible values are shown below:

```
MemoryInGB, 1
NumThreads, 1
OutputFile, auto   [ Possible  values: stdout, quiet, or FileName ]
ScratchDir, auto   [ Possivle values: DirName]
RemoveOutputFile, yes   [ Possible values: yes, no, true, or false]
```

These parameters control the runtime behavior of Psi4.

The default file name for 'OutputFile' is <InFileRoot>_Psi4.out. The PID is appended to output file name during multiprocessing as shown: <InFileRoot>_Psi4_<PIDNum>.out. The 'stdout' value for 'OutputType' sends Psi4 output to stdout. The 'quiet' or 'devnull' value suppresses all Psi4 output.

The default 'Yes' value of 'RemoveOutputFile' option forces the removal of any existing Psi4 before creating new files to append output from multiple Psi4 runs.

The option 'ScratchDir' is a directory path to the location of scratch files. The default value corresponds to Psi4 default. It may be used to override the deafult path.

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

A comma delimited list of parameter name and value pairs for visualizing cube files in PyMOL.

```
contourColor1, red, contourColor2, blue,
contourLevel1, auto, contourLevel2, auto,
contourLevelAutoAt, 0.5,
displayMolecule, BallAndStick, displaySphereScale, 0.2,
displayStickRadius, 0.1, hideHydrogens, yes,
meshWidth, 0.5, meshQuality, 2,
surfaceQuality, 2, surfaceTransparency, 0.25,
volumeColorRamp, auto, volumeColorRampOpacity,0.2
volumeContourWindowFactor,0.05
```

contourColor1 and contourColor2: Color to use for visualizing volumes, meshes, and surfaces corresponding to the negative and positive values in cube files. The specified values must be valid PyMOL color names. No validation is performed.

contourLevel1 and contourLevel2: Contour levels to use for visualizing volumes, meshes, and surfaces corresponding to the negative and positive values in cube files. Default: auto. The specified values for contourLevel1 and contourLevel2 must be negative and positive numbers.

The contour levels are automatically calculated by default. The isocontour range for specified percent of the density is retrieved from the cube files. The contour levels are set at 'contourLevelAutoAt' of the absolute maximum value of the isocontour range. For example: contour levels are set to plus and minus 0.03 at 'contourLevelAutoAt' of 0.5 for isocontour range of -0.06 to 0.06 covering specified percent of the density.

contourLevelAutoAt: Set contour levels at specified fraction of the absolute maximum value of the isocontour range retrieved from the cube files. This option is only used during the automatic calculations of the contour levels.

displayMolecule: Display mode for molecules. Possible values: Sticks or BallAndStick. Both displays objects are created for molecules.

displaySphereScale: Sphere scale for displaying molecule during BallAndStick display.

displayStickRadius: Stick radius for displaying molecule during Sticks and BallAndStick display.

hideHydrogens: Hide hydrogens in molecules. Default: yes. Possible values: yes or no.

meshQuality: Mesh quality for meshes to visualize cube files. The higher values represents better quality.

meshWidth: Line width for mesh lines to visualize cube files.

surfaceQuality: Surface quality for surfaces to visualize cube files. The higher values represents better quality.

surfaceTransparency: Surface transparency for surfaces to visualize cube files.

volumeColorRamp: Name of a PyMOL volume color ramp to use for visualizing cube files. Default name(s): <OutfielsMolPrefix>_psi4_cube_dual or psi4_cube_dual The default volume color ramps are automatically generated using contour levels and colors during 'auto' value of 'volumeColorRamp'. An

explicitly specified value must be a valid PyMOL volume color ramp. No validation is preformed.

VolumeColorRampOpacity: Opacity for generating volume color ramps for visualizing cube files. This value is equivalent to 1 minus Transparency.

volumeContourWindowFactor: Fraction of contour level representing window widths around contour levels during generation of volume color ramps for visualizing cube files. For example, the value of 0.05 implies a ramp window size of 0.0015 at contour level of 0.03.

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

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

-r, --reference <text> [default: auto]

Reference wave function to use for calculating single point energy before generating cube files for dual descriptors corresponding to frontier molecular orbitals. Default: RHF or UHF. The default values are Restricted Hartree-Fock (RHF) for closed-shell molecules with all electrons paired and Unrestricted artree-Fock (UHF) for open-shell molecules with unpaired electrons.

The specified value must be a valid Psi4 reference wave function. No validation is performed. For example: ROHF, CUHF, RKS, etc.

The spin multiplicity determines the default value of reference wave function for input molecules. It is calculated from number of free radical electrons using Hund's rule of maximum multiplicity defined as 2S + 1 where S is the total electron spin. The total spin is 1/2 the number of free radical electrons in a molecule. The value of 'SpinMultiplicity' molecule property takes precedence over the calculated value of spin multiplicity.

-w, --workingdir <dir>

Location of working directory which defaults to the current directory.

EXAMPLES

To generate and visualize dual descriptors from frontier orbitals based on a single point energy calculation using B3LYP/6-31G** and B3LYP/6-31+G** for non-sulfur and sulfur containing closed-shell molecules in a SD file with 3D structures, and write a new PML file, type:

```
% Psi4VisualizeDualDescriptors.py -i Psi4Sample3D.sdf
  -o Psi4Sample3DOut.pml
```

To run the first example to only generate cube files and skip generation of a PML file to visualize dual descriptors for frontier molecular orbitals, type:

```
% Psi4VisualizeDualDescriptors.py --mode GenerateCubeFiles
  -i Psi4Sample3D.sdf -o Psi4Sample3DOut.pml
```

To run the first example to skip generation of cube files and use existing cube files to visualize dual descriptors for frontier molecular orbitals and write out a PML file, type:

```
% Psi4VisualizeDualDescriptors.py --mode VisualizeCubeFiles
  -i Psi4Sample3D.sdf -o Psi4Sample3DOut.pml
```

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

```
% Psi4VisualizeDualDescriptors.py --mp yes -i Psi4Sample3D.sdf
    -o Psi4Sample3DOut.pml
```

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

```
% Psi4VisualizeFrontierOrbitals.py  --mp yes --mpParams "inputDataMode,
    InMemory" -i Psi4Sample3D.sdf  -o Psi4Sample3DOut.pml
```

To run the first example in multiprocessing mode on all available CPUs without loading all data into memory along with multiple threads for each Psi4 run and write out a SD file, type:

```
% Psi4VisualizeDualDescriptors.py --mp yes --psi4RunParams
  "NumThreads,2" -i Psi4Sample3D.sdf -o Psi4Sample3DOut.pml
```

To run the first example in using a specific set of parameters to generate and visualize dual descriptors for

frontier molecular orbitals and write out a PML file, type:

```
% Psi4VisualizeDualDescriptors.py --mode both -m SCF -b aug-cc-pVDZ
  --psi4CubeFilesParams "gridSpacing, 0.2, gridOverage, 4.0"
  --psi4RunParams "MemoryInGB, 2" --pymolViewParams "contourColor1,
  red, contourColor2, blue,contourLevel1, -0.04, contourLevel2, 0.04,
  contourLevelAutoAt, 0.75,volumeColorRamp, auto,
  volumeColorRampOpacity,0.25, volumeContourWindowFactor,0.05"
  -i Psi4Sample3D.sdf -o Psi4Sample3DOut.pml
```

AUTHOR

Manish Sud(msud@san.rr.com)

SEE ALSO

Psi4PerformMinimization.py, Psi4GenerateConformers.py, Psi4VisualizeElectrostaticPotential.py , Psi4VisualizeFrontierOrbitals.py

COPYRIGHT