CalculatePartialChargesWithPsi4

December 3, 2021

0.1 Calculate partial charges with Psi4

import the following modules

[1]:
```
import os
import psi4
import resp
import openbabel as ob
from rdkit import Chem
from rdkit.Chem import AllChem
```

0.1.1 some helper functions

[2]:
```
def neutralize_atoms(mol):
    pattern = Chem.MolFromSmarts("[+1!h0!$([*]~[-1,-2,-3,-4]),-1!
 ↪$([*]~[+1,+2,+3,+4])]")
    at_matches = mol.GetSubstructMatches(pattern)
    at_matches_list = [y[0] for y in at_matches]
    if len(at_matches_list) > 0:
        for at_idx in at_matches_list:
            atom = mol.GetAtomWithIdx(at_idx)
            chg = atom.GetFormalCharge()
            hcount = atom.GetTotalNumHs()
            atom.SetFormalCharge(0)
            atom.SetNumExplicitHs(hcount - chg)
            atom.UpdatePropertyCache()
    return mol

def cleanUp(psi4out_xyz):
    deleteTheseFiles = ['1_default_grid.dat','1_default_grid_esp.dat','grid.
 ↪dat','timer.dat']
    deleteTheseFiles.append(psi4out_xyz)
    for fileName in deleteTheseFiles:
        if os.path.exists(fileName):
            os.remove(fileName)

def get_xyz_coords(mol):
    if not mol is None:
        num_atoms = mol.GetNumAtoms()
        xyz_string=""
        for counter in range(num_atoms):
            pos=mol.GetConformer().GetAtomPosition(counter)
            xyz_string = xyz_string + ("%s %12.6f %12.6f %12.6f\n" % (mol.
 ↪GetAtomWithIdx(counter).GetSymbol(), pos.x, pos.y, pos.z) )
    return xyz_string


def calcRESPCharges(mol, basisSet, method, gridPsi4 = 1):
    options = {'BASIS_ESP': basisSet,
               'METHOD_ESP': method,
               'RESP_A': 0.0005,
               'RESP_B': 0.1,
               'VDW_SCALE_FACTORS':[1.4, 1.6, 1.8, 2.0],
               'VDW_POINT_DENSITY':int(gridPsi4)
    }

    resp_charges = resp.resp([mol], [options])[0][1]
    return resp_charges
```

0.1.2 Set some variables and stuff

[3]:
```
method = 'b3lyp'
basisSet = '3-21g'
neutralize = True
psi4.set_memory('10 GB')
obConversion = ob.OBConversion()
obConversion.SetInAndOutFormats("xyz", "mol2")
singlePoint = True
path = "./data"
```

0.1.3 Read sdf file (3D) into a list

[4]:
```
inputFile = "./data/twoCpds.sdf"
molList = Chem.SDMolSupplier(inputFile)#, removeHs=False)
```

0.1.4 Loop over compounds in list and calculate partial charges

[7]:
```
reportFile = "psi4report.csv" # results here
outFile = open(reportFile,'w')

for mol in molList:
    if not mol is None:

        #print('before', Chem.MolToSmiles(mol))
        if neutralize:
            mol = neutralize_atoms(mol)
            mol = Chem.AddHs(mol)
            AllChem.EmbedMolecule(mol,randomSeed=0xf00d)
            #print('after',Chem.MolToSmiles(mol))

        xyz_string = get_xyz_coords(mol)
        psi_mol = psi4.geometry(xyz_string)

        ### single point calculation
        molId = mol.GetProp("_Name")
        outfile_mol2 = inputFile[:-4]+".mol2"

        if singlePoint:
            print('Running singlepoint', molId)
            resp_charges = calcRESPCharges(psi_mol, basisSet, method, gridPsi4␣
↪= 1)
            #resp_charges = calcRESPCharges(psi_mol, basisSet = '6-31G**',␣
↪method = 'b3lyp', gridPsi4 = 1)

        else:
            ### geometry optimization
            methodNbasisSet = method+"/"+basisSet
            psi4.optimize(methodNbasisSet, molecule=psi_mol)
            resp_charges = calcRESPCharges(psi_mol, basisSet, method, gridPsi4␣
↪= 1)
            #psi4.optimize('b3lyp/6-31g**', molecule=psi_mol)

        ### save to xyz file
        psi4out_xyz = molId + '.xyz'
        psi_mol.save_xyz_file(psi4out_xyz,1)

        ### read back (xyz) and write as mol2
        ob_mol = ob.OBMol()
        obConversion.ReadFile(ob_mol, psi4out_xyz)

        ### set new charges back
        count = 0
        for atom in ob.OBMolAtomIter(ob_mol):
            newChg = resp_charges[count]
            atom.SetPartialCharge(newChg)
            count += 1

        ### write as mol2
        outfile_mol2 = path+"/"+molId+"_partialChgs.mol2"
        print("Finished. Saved compound with partial charges as mol2 file: %s"␣
↪% outfile_mol2)
```

```
        obConversion.WriteFile(ob_mol, outfile_mol2)

        ### clean up
        cleanUp(psi4out_xyz)

outFile.close()
```

```
running singlepoint AMHA90
Finished. Saved compound with partial charges as mol2 file:
./data/AMHA90_partialChgs.mol2
running singlepoint 4piol
Finished. Saved compound with partial charges as mol2 file:
./data/4piol_partialChgs.mol2
```

[ ]:

```

```