Instructions and explanation of implementation of python's pyvoro package for hydration analysis

Madeline Galbraith
Madura Research Group

November 18, 2015

1 Explanation

Note: This code works only for wrapped boxes in which the protein/polymer has been centered

- The entire program uses python's MDAnalysis package
- Voronoi tesselations are created using the pyvoro package, a wrapper class for the Voro++ library in C++
- analysis.py is the main program
  - All classes and functions are accessed through this file.
- myFiles.py has all of the information for input and initializes the output files
- Selections.py has all of the selection commands used in the input
- Weights.py holds the weighting options
  - They are used in the creation of radical tesselations
- Waters.py calculates the number of waters within the specified distance and the tetrahedrality parameter
- pyvoroTesselations uses the pyvoro package to calculate hydration properties(volumes, hydration shells, and PMV)
  - All voronoi polyhedra calculates are done in this class/file

2 Instructions

NOTE: You need all python files in the folder (i.e. analysis.py, myFiles.py, pyvoroTesselations.py, Selections.py, Waters.py, Weights.py)

- In **myFiles.py**
  1. Change line 16 to be filename =“file.pdb” or filename =“file.psf”
  2. Change line 22 to be the rootname.
  3. Modify lines 25 to 30 to add all the files you want to analyze to the filelist
- In **Selections.py**
  1. Create a new method that returns the selection
     - def **getThisSelection**(self):
     - return ”selection” #this is according to MDAnalysis selection commands(for examples see end)
- In **Weights.py**
  1. This class only computes weights based upon VanDerWaalsWeighting at the moment
  2. Add any elements that are present in the system, but not in the class according to the example
     - Use the element symbol and Van der Waals radii

- **Waters.py** is an analysis file
  - Do not modify
- **pyvoroTesselations.py** is an analysis file
  - Tesselation created using command:
    - vor = pyvoro.compute_voronoi(coordinates,limits, 2.0, radii = weights, periodic = [True, True, True])
    - coordinates = the atoms in the box in the form [[x,y,z],[x,y,z]]
    - limits = box limits
    - 2.0 = how far apart the atoms can be for interaction and tesselation to be created
    - radii = the weights to be used must be in the same pattern as the coordinates input (i.e. coordinates = [oxygen, hydrogen, carbon, oxygen] radii = [oxygen, hydrogen, carbon, oxygen])
    - periodic = periodic in x, periodic in y, periodic in z
  - Do not modify
- In **analysis.py**
  1. Change line 53 to be selectionOfinterest = Selections.Selections().**yourSelectionMethodName**()
  2. Lines 72-84 can be commented out or used to calculate number of waters around a particular group do :
     - groupselection = Selections.Selections().**yourSelectionMethodName**(universe)
     - group = universe.selectAtoms(groupselection)
     - groupWaters = Waters.Waters()
  3. Change line 90 this will be the start, end and step for your trajectory :
     - for ts in universe.trajectory[start:end:step]:
  4. Lines 98-111 can be commented out or used to calculate number of waters and tetrahedrality around the particular groups do:
     - groupWaters.getnumWaters(selectionOfWater, group,universe, **distance from waters**)
     - groupWaters.printToTable(”yourFileName.dat”,ts.frame, groupWaters.numOfWater)
     - groupWaters.tetrahedrality(moleculesOfWater, ts, BoxForWaters)
     - groupWaters.printToTable(”yourFileName.dat”,ts.frame, groupWaters.SgParameter)

3 Visualizing Hydration Shells

- Open VMD and load the frame the analysis.py went through given in Frame.dat
  1. go to Graphics then Representations
  2. in the selected atoms box type in ”index ” then copy and paste the values from the FirstShellIndices_VMD.dat to view hydration shell one
  3. press create rep
  4. repeat b-c for values from SecondShellIndices_VMD.dat to view hydration shell two

4 Understanding the output files

Files: Note that all files have headers.

- The following four files give the volume at each timestep (and the last two also give the number of waters in the hydration shell)

  1. PMV.dat
     - column1 column2
     - frame partial molar volume
  2. mainMoleculeVolume.dat
     - column1 column2
     - frame volume for molecule of interest
  3. FirstHydrationShell.dat
     - column1 column2
     - frame total volume of first hydration shell number of waters
  4. SecondHydrationShell.dat
     - column1 column2
     - frame total volume of second hydration shell number of waters

- The following three files give the number of waters within (3Å,3Å,4.5Å) of each monomer unit

  - column1 column2-30
  - frame number of waters within specified distance of functional group in this monomer

  1. AmideNumberOfNearWaters.dat
  2. CarbonylNumberOfNearWaters.dat
  3. RgroupNumberOfNearWaters.dat

  - column1 column2-30
  - frame tetrahedrality of surrounding waters of functional group in this monomer

  1. Amide_Tetrahedrality.dat
  2. Carbonyl_Tetrahedrality.dat
  3. Rgroup_Tetrahedrality.dat

- The following three files are needed to visualize the hydration shells.

  1. Frame.dat
     - Tells you the frame used to generate the following two files
  2. FirstShellIndices_VMD.dat
     - Atom indices for water oxygens in the first hydration shell
  3. SecondShellIndices_VMD.dat
     - Atom indices for water oxygens in the second hydration shell

- The following four files give the index and volume of each atom or molecule in the selection(at a time that is an average).

  - column1 column2
  - index volume

  1. mainMoleculeVolumesAtAvg.dat
  2. FirstShellVolumesAtAvg.dat
  3. SecondShellVolumesAtAvg.dat
  4. BulkVolumesAtAvg.dat

5 List of MDAnalysis Selection Commands

NOTE: This list is not exhaustive and more can be found online (such as geometric selections). All must be in string format.
Boolean characters are valid.

- ”bynum index”, ”bynum starting index:ending index”
- ”name ” (specified in file)
- ”type” (specified in file)
- ”segid” segname
- ”resid” residue number range
- ”resnum” resnum number name
- ”resname” residue name