An Open-Source Python Toolkit for Interactive and High-Throughput Analysis of Solvation Structures

Orion Cohen[1], Hugo Macdermott-Opeskin[2], Kara Fong[1], Tingzheng Hou[1], Kristin Persson[1, 3]

[1]University of California Berkeley [2]Australian National University [3]Molecular Foundry

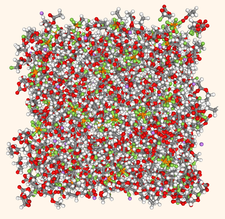
Abstract

Molecular dynamics **studies of liquid solvation structures often replicate established analyses on novel systems**. In electrolyte systems, it is common to calculate coordination numbers, radial distribution functions, solute dissociation, cluster speciation, etc. In principle, these analyses are highly similar across a diversity of systems. In practice, many specialized bespoke tools have sprung up to address the same underlying problem. **Enter Solvation Analysis, an easy-to-use Python package with an interactive interface for computing a wide variety of solvation properties**. Building on MDAnalysis and Pandas, it efficiently processes output from a wide variety of Molecular Dynamics applications. Paired with high-throughput infrastructure, Solvation Analysis becomes a powerful tool for understanding solvation at scale.

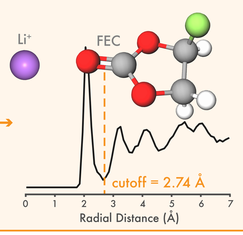
```python
import MDAnalyis

u = MDAnalysis.Universe("path_to_topology", "path_to_trajectory")

Li = u.atoms.select_atoms("element Li")
PF6 = u.atoms.select_atoms("byres element P")
EA = u.atoms.select_atoms("resnum 0:234")
FEC = u.atoms.select_atoms("resnum 235:600")
```

powered by MDAnalysis

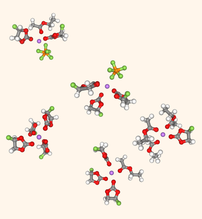
Full System Imported



Solvation Cutoff Detected



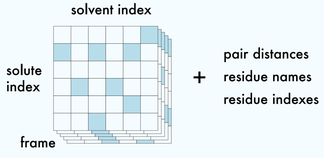
Solvation Shells Isolated



```python
from solvation_analysis import Solution

solution = Solution(li_atoms, {'PF6': PF6, 'EA': EA, 'FEC': FEC})
solution.run()
print(solution.solvation_data)
```

frame	solvated atom	atom ix	dist	res name	res ix
0	0	4800	1.96	FEC	386
		1111	1.98	EA	79
		6890	2.17	FEC	595
		5270	2.20	FEC	433
...	...	...	...	...	...
		3330	2.14	FEC	239
		3350	2.15	FEC	241
		4570	2.29	FEC	363
		4460	2.30	PF6	352

Solvation Data is a Sparse Adjacency Matrix



Networking

Study the topology and structure of solute-solvent networks.

Networking yields a complete description of coordinated solute-solvent networks in the solution, regardless of identify. This could include cation-anion networks or hydrogen bond networks.

Speciation

Explore the precise solvation shell of every solute.

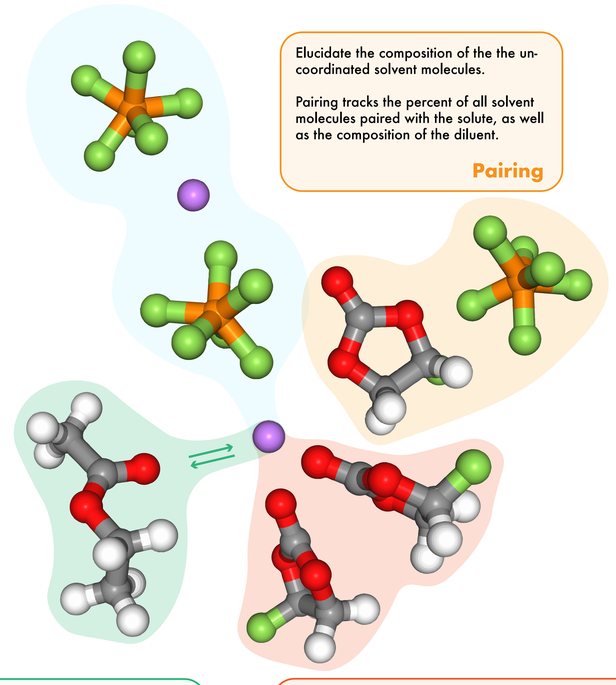
Speciation tabulates the unique solvation shell compositions, their percentage, and their temporal locations.

From this, it provides search functionality to query for specific solvation shell compositions. Extremely convenient for visualization.

Pairing

Elucidate the composition of the the uncoordinated solvent molecules.

Pairing tracks the percent of all solvent molecules paired with the solute, as well as the composition of the diluent.



Residence

Understand the dynamic coordination of solvents with the solute.

Residence times for all solvents are automatically calculated from autocovariance of solvent-solute adjacency matrix.

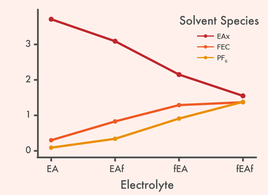
Coordination

Elucidate the coordination of each solvating species.

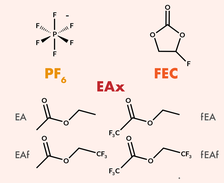
Coordination numbers for each solvent are automatically calculated, along with the types of every coordinating atom.

```python
# coming soon
from solvation_analysis import plotting

# define each solution as shown previously
solutions = [solution_ea, solution_eaf, solution_fea, solution_feaf]

plotting.plot_coordination_numbers(solutions).show()
plotting.plot_residence_times(solutions).show()
```

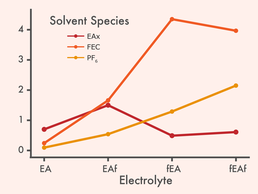
Coordination Number



Here, solvation structure shifts with increasing co-solvent fluorination.



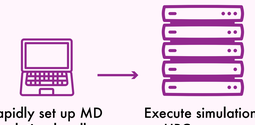
Residence Time (ns)



Coming soon, automated plotting of all solvation information. The plotting module will support a rapid, intuitive understanding of solvation, both within a system and between systems.

```python
from pymatgen.io.openmm import OpenMMSolutionGen

openmm_generator = OpenMMSolutionGen(force_field="amber")

electrolyte = {"CCOC(=O)C": 200, "F[P-](F)(F)(F)(F)F": 25, "[Li+]": 25}
input_set = openmm_generator.get_input_set(electrolyte, density=1.34)

simulation = input_set.get_simulation()  # ready for any workflow
```

High-Throughput MD

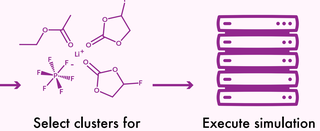


Rapidly set up MD simulation locally.

Execute simulation on HPC resource.

Automate analysis, store in database.

Integration with DFT



Select clusters for DFT simulation.

Execute simulation on HPC resource.

Acknowledgements

MDAnalysis Dev Team
Oliver Beckstein
Richard Gowers
Irfan Alibay
Lily Wang

Persson Group
Kristin Persson
Ryan Kingsbury
Jingyang Wang
Lauren Lee

Funding
Google Summer of Code
NSF GRFP Fellowship
US Department of Energy

Software
MDAnalysis
Pymatgen
Pandas
NGLView