LOOS: A Tool for Making New Tools for Analyzing Molecular Simulations

Tod D. Romo, Akshara Sharma, Louis G. Smith,† and Alan Grossfield

University of Rochester Medical School, Rochester, NY, USA

†University of Pennsylvania School of Medicine, Philadelphia, PA, USA

Analyzing Molecular Simulations

- Molecular simulations have unmatched resolution in time and space
- Need better analysis tools to extract maximum value
  - Most projects require custom code
  - Data analysis is an interative process
  - Rapid development is key

LOOS Design Goals

Package-agnostic

- Read all common file formats
  - NAMD, Amber, GROMACS, TINKER, OpenMM, LAMMPS, MDTraj
- Programs don't care where files came from
- Reduce duplicated effort
- Improve reproducibility

Easy to use

- Powerful tools
- Unique functionality
- Convenient atom selection facility
- Highly scriptable
- Detailed documentation
- High performance
  - 1-2 order of magnitude faster than VMD and MDAnalysis
  - Comparable to cpptraj

Easy to develop

- C++ core
  - Good object design makes code expressive
  - 4 key classes means easy to learn
  - No memory management
  - Atom selection identical to tool level
- Python interface
  - Rapid application development
  - Easy code reuse
  - Interoperable with NumPy, SciPy, Scikit-learn
- Extensive documentation
  - Code level via doxygen
  - Github wiki

Available everywhere

- Install via conda-forge or from source
- Linux and Mac
- Windows via WSL

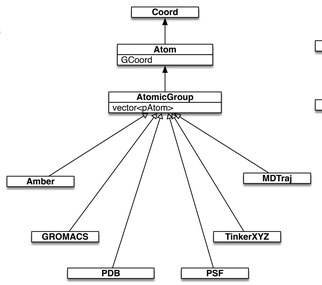
Using LOOS

Example Tools

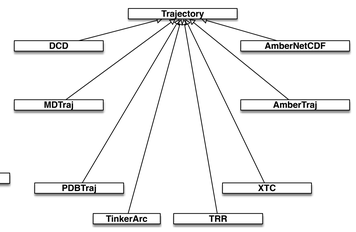
Macromolecules	
rmsds	All-to-all structure comparison
rmsf	Molecular fluctuations
svd	Principal component analysis
rdf	Radial distribution function
Membrane and Membrane Proteins	
order_parameters	Chain tilt in membrane
density-dist	Distribution along membrane normal
xy_rdf	Radial distribution in membrane plane
membrane_map	Lipid properties around membrane
mops	Molecular order parameters for chains
dibmops	mops vs distance from macromolecule
Trajectory Manipulation	
merge-traj	Rapidly merge and manipulate trajectories
subsetter	Merge, reimage, and subset trajectories
Convergence Package	
block_average	Block average of time-series data
decorr_time	Decorrelation time of structural histograms
bcom, boot_mbcom	Block covariance overlap method
Voronoi Package	
area_per_molecule.py	Area distribution for a membrane slice
area_profile.py	Voronoi area for protein along normal
Elastic Network Model Package	
anm	Anisotropic network model
enmovie	Visualize ENM modes via a trajectory
vsa	Vibrational subsystem analysis
Gridded Density Package	
water-hist	3D density histograms for atoms
near_blobs	Find residues near density peaks
grid2xplor	Convert density grid to X-plor format for visualization
Optimal Membrane Generator Package	
omg.py	Build membrane systems
solvate.py	Build water around soluble molecules

Developing with LOOS

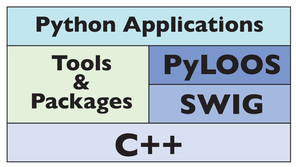
Structure Classes



Trajectories



Application Layers



Classes map to key concepts

- Makes code expressive
- AtomicGroup is workhorse
  - File formats are subclasses
  - Common tasks are methods
- Trajectory formats are subclasses
- Code with parent classes is format-agnostic

Most new tools are python

- Easier development
- Great libraries
  - Interoperate via NumPy, SciPy, Scikit-learn
  - Voronoi package, clustering
- Performance gap is small
- Most work is done in the library

Example Code

protein_tilt.py

```python
#!/usr/bin/env python3

import loos
import loos.pyloos
import sys
import math

if len(sys.argv) < 4 or sys.argv[1] == "-h" or sys.argv[1] == "--help":
    print("Usage: ", sys.argv[0], " system trajectory selection1 [selection2...]")
    print("       Prints the average of the orientation vectors of the ")
    print("       individual selections, assuming each individual vector ")
    print("       points in the +z direction")
    sys.exit()

print("#", " ".join(sys.argv))
system_filename = sys.argv[1]
traj_filename = sys.argv[2]
selections = sys.argv[3:]

system = loos.createSystem(system_filename)
traj = loos.pyloos.Trajectory(traj_filename, system)

helices = []
for s in selections:
    helices.append(loos.selectAtoms(system, s))

to_degrees = 180.0/math.pi

print("#Frame\tAngle\tCosine")

for _ in traj:

    vec = loos.GCoord(0.0, 0.0, 0.0)
    for h in helices:
        pca = h.principalAxes()
        v = pca[0]
        if v.z() < 0:
            v *= -1.0
        vec += v

    cosine = vec.z() / vec.length()

    cosine = max(-1.0, cosine)
    cosine = min(1.0, cosine)
    ang = math.acos(cosine) * to_degrees

    print(traj.index(), ang, cosine)
```

Future Directions

- Built-in featurizers for MSMs
- Non-orthonormal periodic boxes
- Better community engagement
- Take over the world..

Getting Help

- Github
  - User and developer docs
  - HowTo stories on GitHub wiki
  - Discussions for strategic conversations
  - Issues for bugs and feature requests
- loos.maintainer@gmail.com

New in LOOS 4.2

- Performance improvements
- Compatible with Numpy 2.0
- Improved conda-forge packaging
- Improved error messages

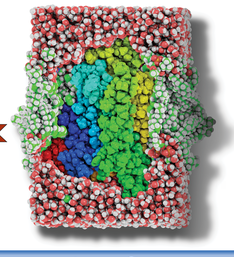
Getting LOOS

LOOS Paper

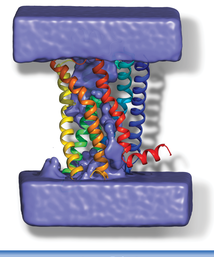
This Poster

LOOS on GitHub

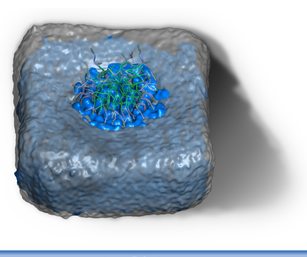
System Visualization



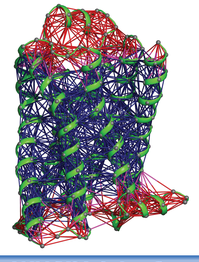
Water Density



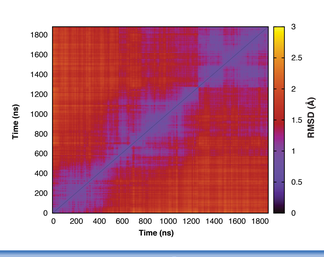
Lipid Density



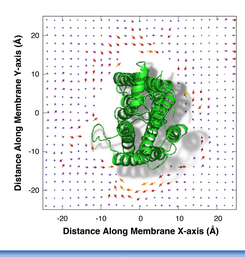
Elastic Network Models



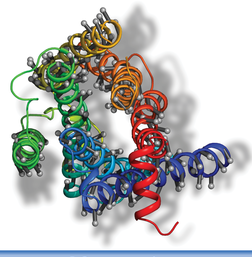
All-to-All RMSD



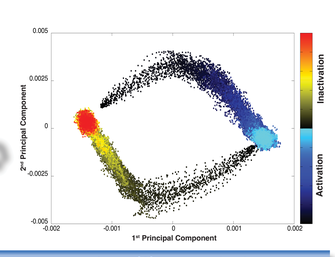
Membrane Properties



Principal Components



Transition Contacts



Tools:	custom software	water-hist	water-hist	ANM/VSA (rebond)	rmsds	membrane_map	svd & porcupine	transitions_contacts