Encoding interactions as fingerprints in MD simulations

Dr. Cédric Bouysset

Université Côte d'Azur, Nice, France

Current affiliation: Exscientia, Oxford, UK

MDAnalysis UGM 2023

Why?

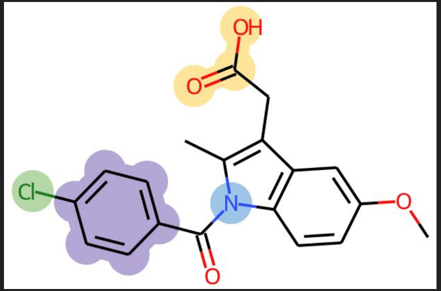
Interactions are the driving force behind biological processes.

Need to automatically detect and label them, and store them in an appropriate way in order to analyse this data and extract information efficiently.

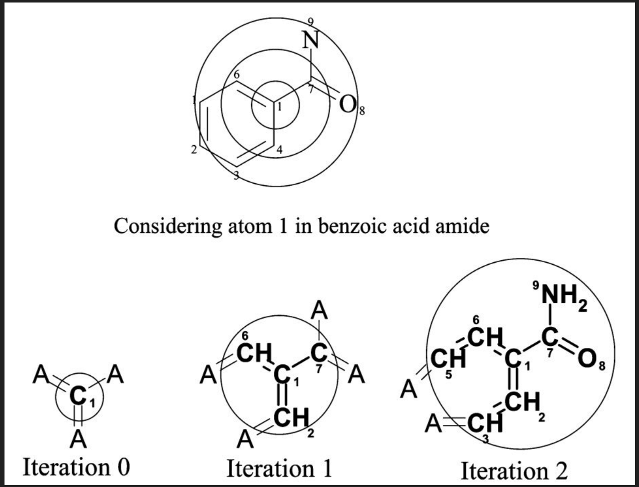
Interaction Fingerprints

Fingerprints in cheminformatics

- Compact representation for small molecules
- Encodes structural information as a bitvector
- Each bit defines the presence or absence of a feature (substructure)
- Different ways to define the substructures
  - predefined: substructure keys, e.g. MACCS
  - automatically generated: circular, e.g. ECFP



0	1	1	0	0	1	0	1



[Rogers and Hahn (2010) J. Chem. Inf. Model.]

Similarity principle

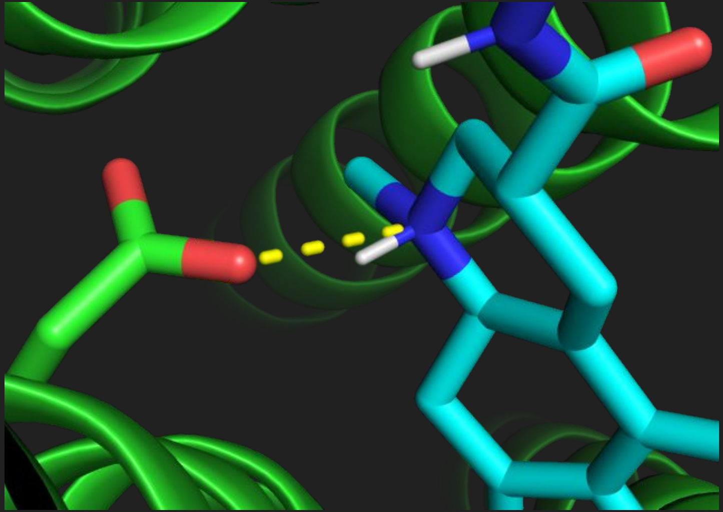
- Structurally similar molecules tend to exhibit similar properties
- Different metrics for similarity between molecules, e.g. Tanimoto

Compound 1	0	1	1	0	0	1	0	1
Compound 2	1	1	0	0	0	1	0	0

$$Tanimoto = \frac{A \cap B}{A \cup B} = \frac{A \cap B}{|A| + |B| - A \cap B} = \frac{2}{4 + 3 - 2} = 0.4$$

Interaction fingerprints?

- Still encodes structural information as a bitvector
- Bitvector that stores the presence/absence of interactions
- Similarity between complexes or binding modes
- Compounds with similar interaction profiles are likely to exhibit similar properties



Flavors of IFP

- Structural IFP
- Pharmacophore IFP
- Circular IFP

Interactions usually defined based on **atom selections** and **geometrical constraints**

Structural IFP

- Each bit position corresponds to an interaction type and residue
- Can only compare IFPs from the same protein family and pocket

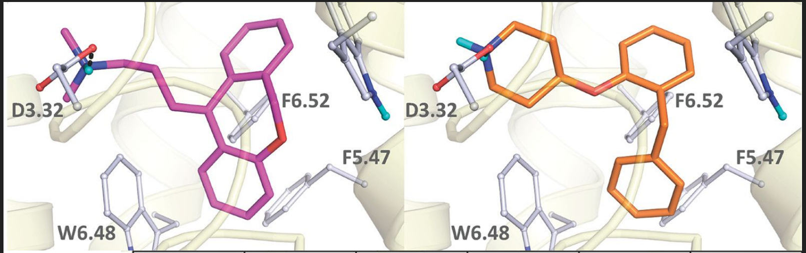
VAL87				ASP114				PHE189				SER193			
Hydrophobic	Acceptor	Donor	PiStacking	Hydrophobic	Acceptor	Donor	PiStacking	Hydrophobic	Acceptor	Donor	PiStacking	Hydrophobic	Acceptor	Donor	PiStacking
1	0	0	0	0	1	0	0	1	0	0	1	0	0	1	0

Deng et al. (2004) J. Med. Chem.

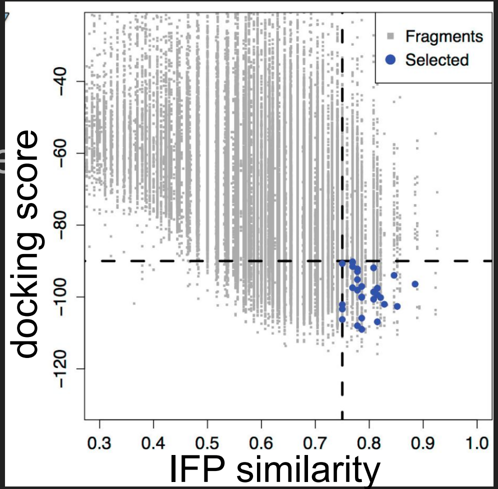
Example use cases

Filtering docking poses

- Human histamine H1 receptor (H1R) GPCR
- Structure with co-crystallized ligand (doxepin)
- Filtered docking poses based on
  - **Presence of ionic interaction with key residue**
  - **IFP-based Tanimoto similarity with reference crystal structure above threshold**
  - Docking score below threshold
  - ECFP4 Tanimoto similarity with any known H1R ligand above threshold
- 19 of 26 compounds tested had affinities in µM to nM range



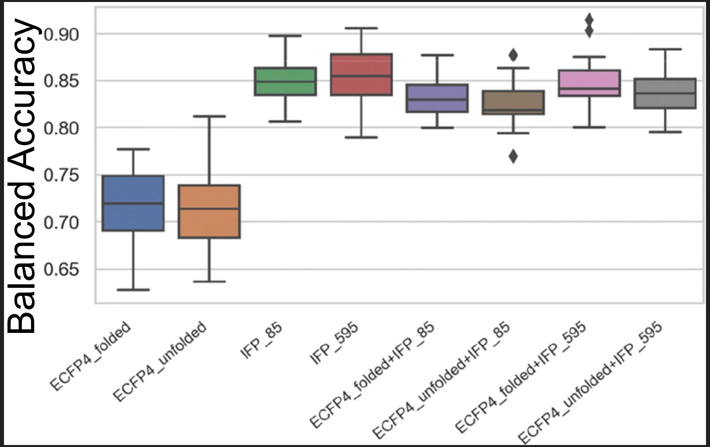
	D3.32	W4.56	F5.47	W6.48	F6.52	F6.55
Doxepin	1000101	1010000	1000000	1010000	1010000	1010000
Compound 3	1000101	1000000	1010000	1010000	1010000	1000000



de Graaf et al. (2011) J. Med. Chem.

Representation for supervised machine-learning models

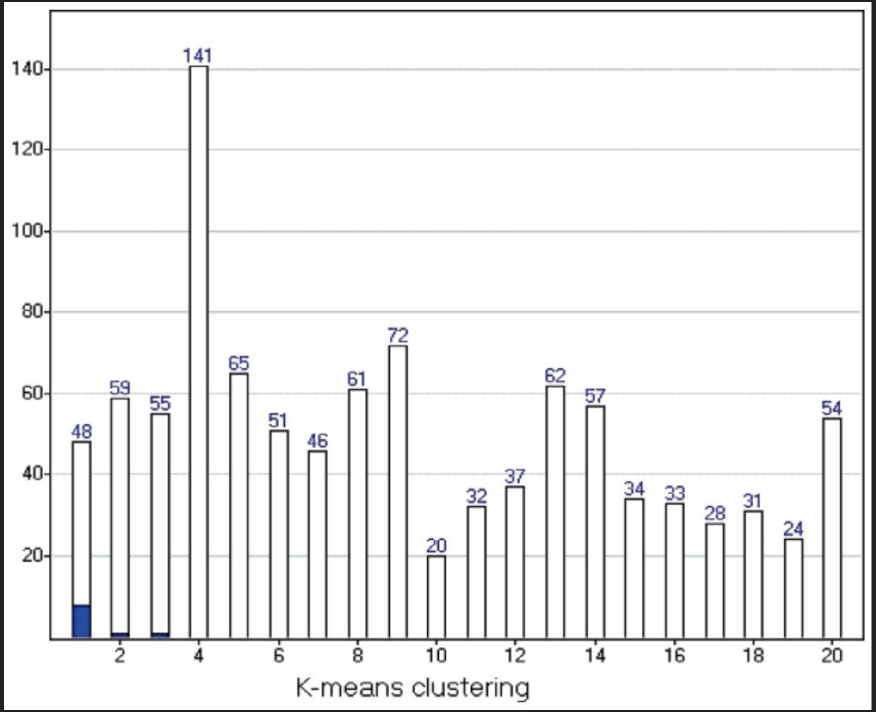
- Kinase inhibitors
- More than 2,000 X-ray structures available
- 3 different binding modes emerge
- Classification task
- IFPs achieved superior predictive performance than 2D-ligand fingerprints (ECFP4)



Rodríguez-Pérez et al. (2020) J. Cheminform.

Extracting binding modes through clustering

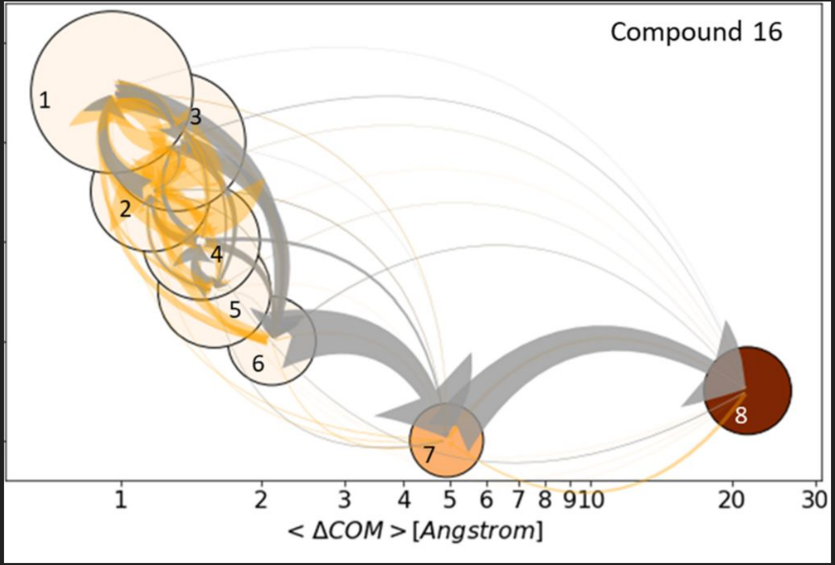
- For compound selection in docking
- Cluster all IFPs using k-means
- Include IFPs of active compounds
- Identify which cluster(s) the actives fall into



Mpamhanga et al. (2006) J. Chem. Inf. Model.

Investigate ligand dissociation pathways

- Biased MD protocol (Random Acceleration MD) to analyse unbinding trajectories
- Extract clusters based on IFPs
- Characterization of the dissociation pathway and metastable states by representing the pathway as a graph



Kokh et al. (2020) J. Chem. Phys.

Limitations of existing IFP implementations

- Only specific files can be read
- Only supports protein-ligand interactions
- Not configurable
- Not extensible
- Poor interoperability with other Python packages

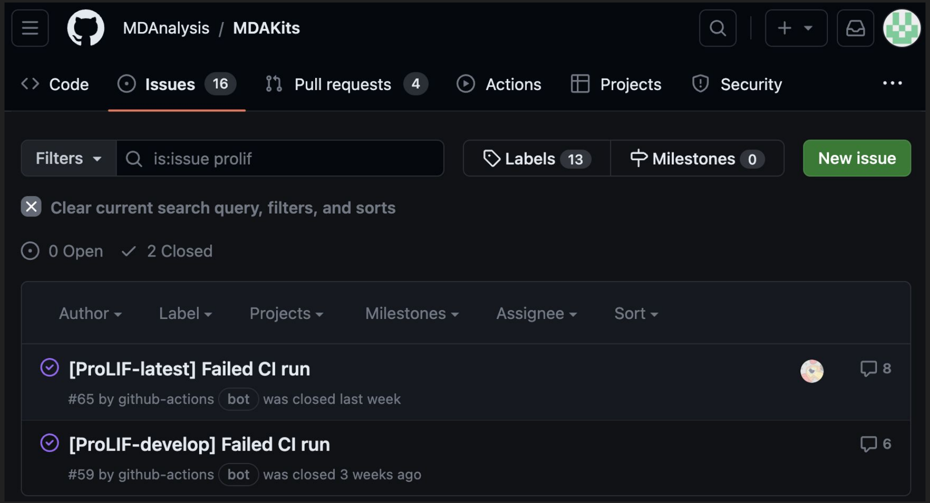
ProLIF

chemosim-lab/ProLIF

Bouysset C. and Fiorucci S. (2021) J. Cheminform.

ProLIF features

- Almost any 3D file can be read
- Supports any kind of complex: protein-ligand, protein-protein…etc.
- Configurable and extensible
- Exports to other Python objects
- Jupyter-notebook centric
- Part of the MDAKits



is:issue prolif

Clear current search query, filters, and sorts

0 Open ✓ 2 Closed

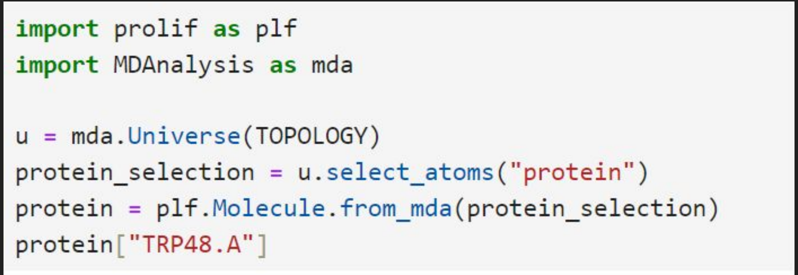
[ProLIF-latest] Failed CI run
#65 by github-actions bot was closed last week 8

[ProLIF-develop] Failed CI run
#59 by github-actions bot was closed 3 weeks ago 6

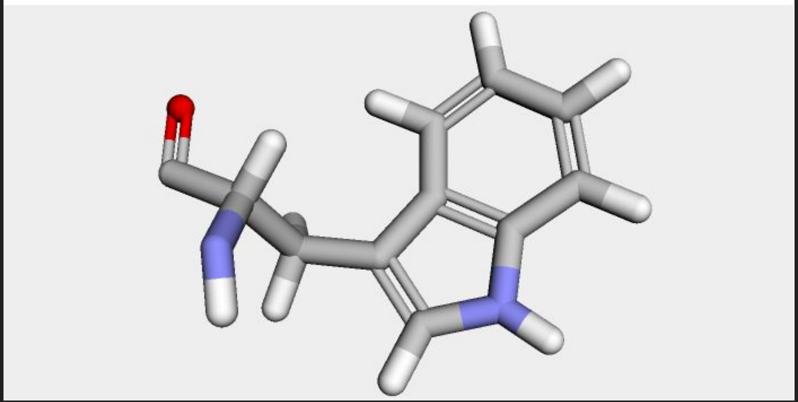
The implementation

Reading inputs

- Uses both MDAnalysis and RDKit
- Internally uses RDKit to represent molecules, with a few extras:
  - Fragment parent molecule into its residues
  - Residue are then available as individual molecules by indexing

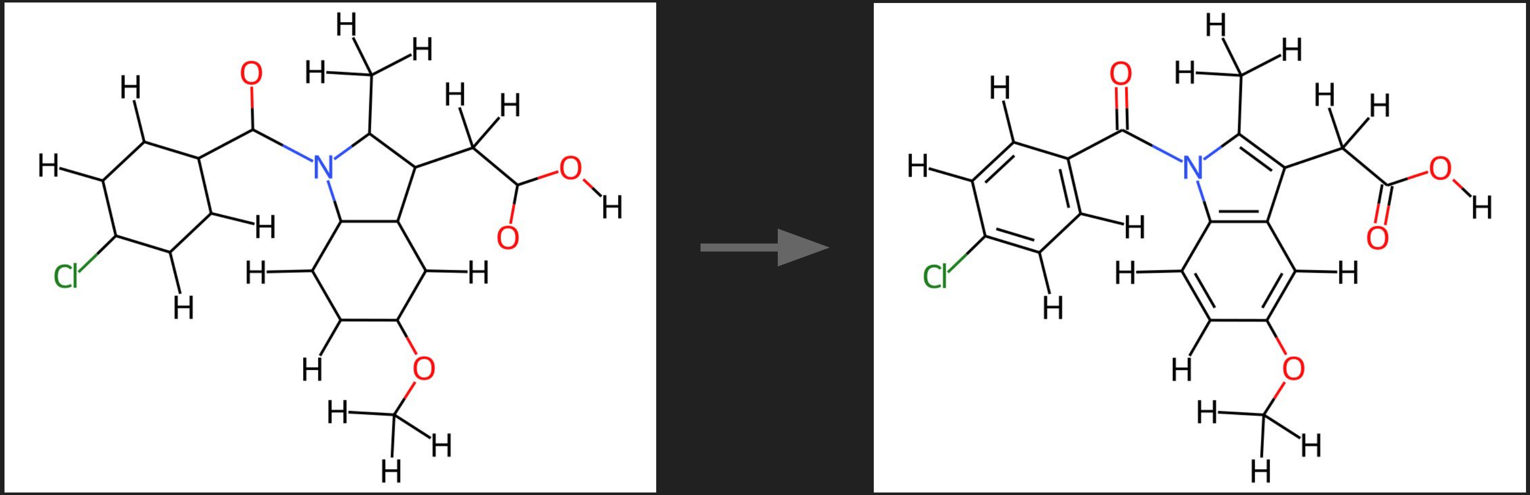


```python
import prolif as plf
import MDAnalysis as mda

u = mda.Universe(TOPOLOGY)
protein_selection = u.select_atoms("protein")
protein = plf.Molecule.from_mda(protein_selection)
protein["TRP48.A"]
```

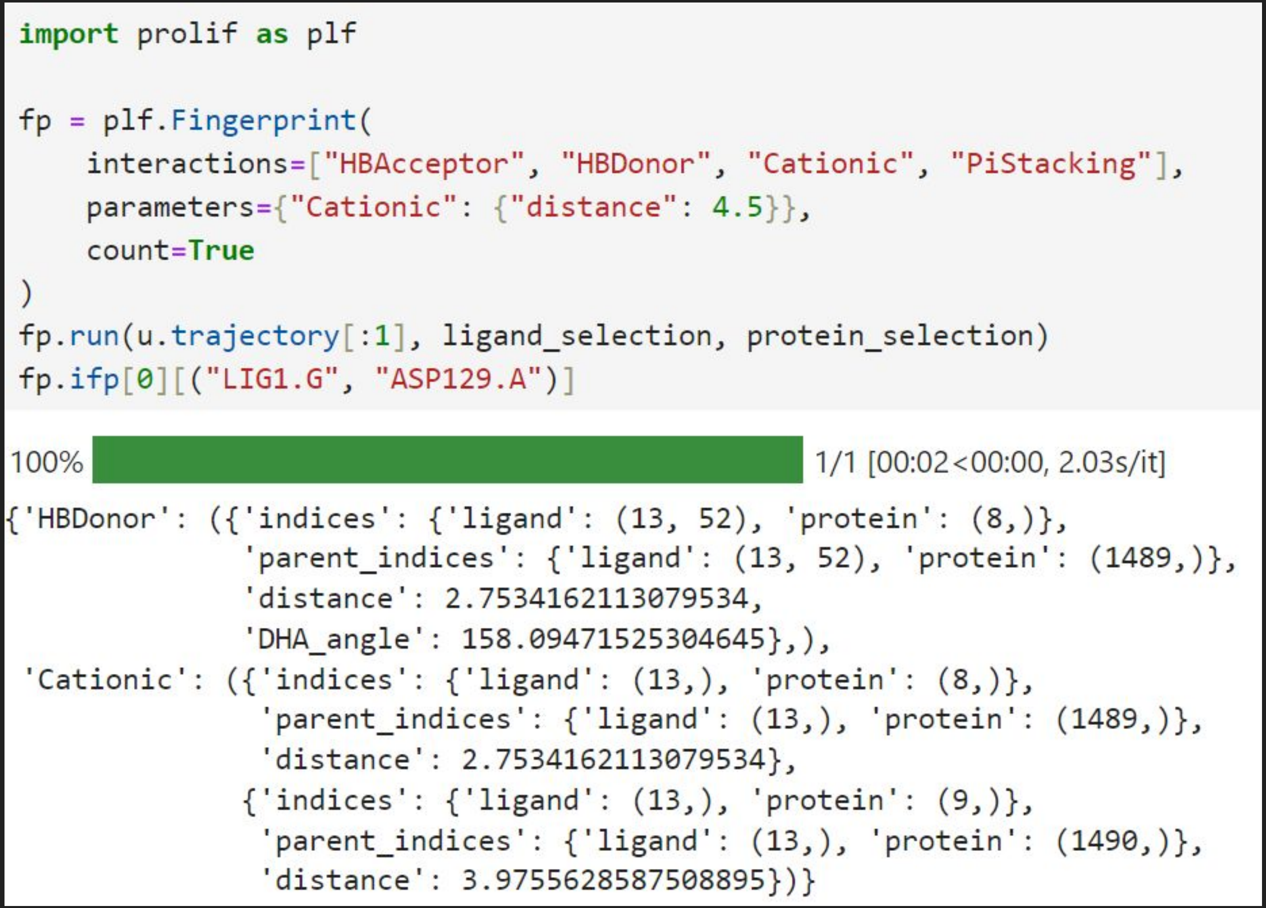


Reading inputs - MD trajectories and PDB files

- Parse file(s) with MDAnalysis
- Convert to an RDKit mol through the converter interface
  - Bond orders and charges have to be inferred from the topology
- Requires explicit hydrogen atoms



Fingerprint

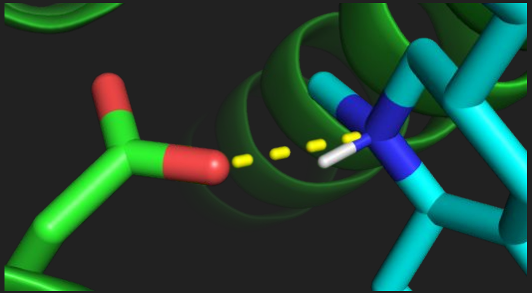


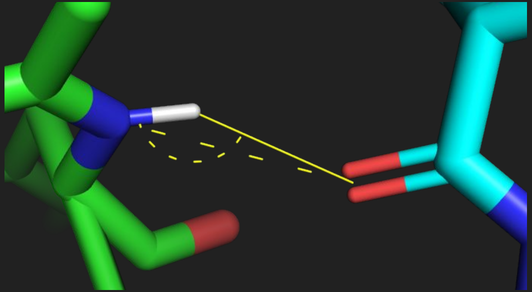
```python
import prolif as plf

fp = plf.Fingerprint(
    interactions=["HBAcceptor", "HBDonor", "Cationic", "PiStacking"],
    parameters={"Cationic": {"distance": 4.5}},
    count=True
)
fp.run(u.trajectory[:1], ligand_selection, protein_selection)
fp.ifp[0][("LIG1.G", "ASP129.A")]
```

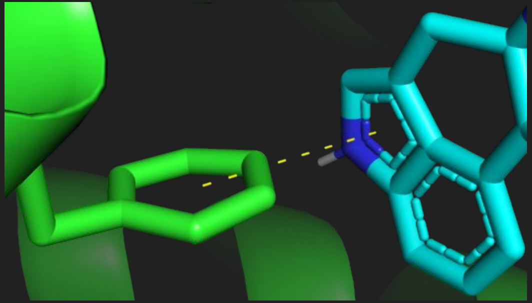
```
100% 1/1 [00:02<00:00, 2.03s/it]
{'HBDonor': ({'indices': {'ligand': (13, 52), 'protein': (8,)},
              'parent_indices': {'ligand': (13, 52), 'protein': (1489,)},
              'distance': 2.7534162113079534,
              'DHA_angle': 158.09471525304645},),
 'Cationic': ({'indices': {'ligand': (13,), 'protein': (8,)},
               'parent_indices': {'ligand': (13,), 'protein': (1489,)},
               'distance': 2.7534162113079534},
              {'indices': {'ligand': (13,), 'protein': (9,)},
               'parent_indices': {'ligand': (13,), 'protein': (1490,)},
               'distance': 3.9755628587508895})}
```

Interactions

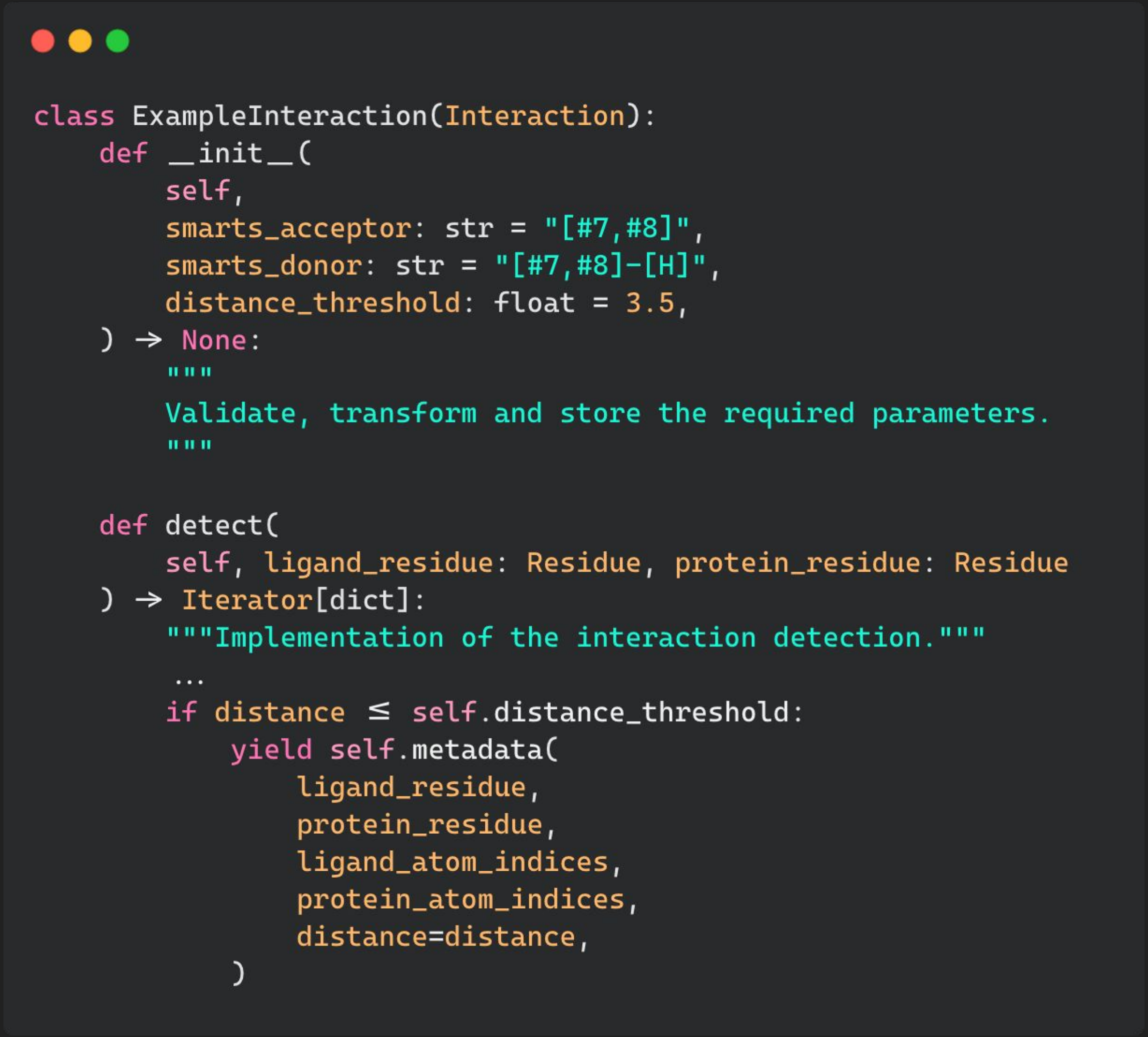
- Currently 15 interactions implemented
- 3 base interaction classes:
  - **Distance**
    - Hydrophobic, Cationic, Anionic, MetalDonor, MetalAcceptor
  - **SingleAngle**
    - HBDonor, HBAcceptor
  - **DoubleAngle**
    - XBDonor, XBAcceptor
- Others:
  - PiStacking, EdgeToFace and FaceToFace
  - PiCation, CationPi
  - VdWContact





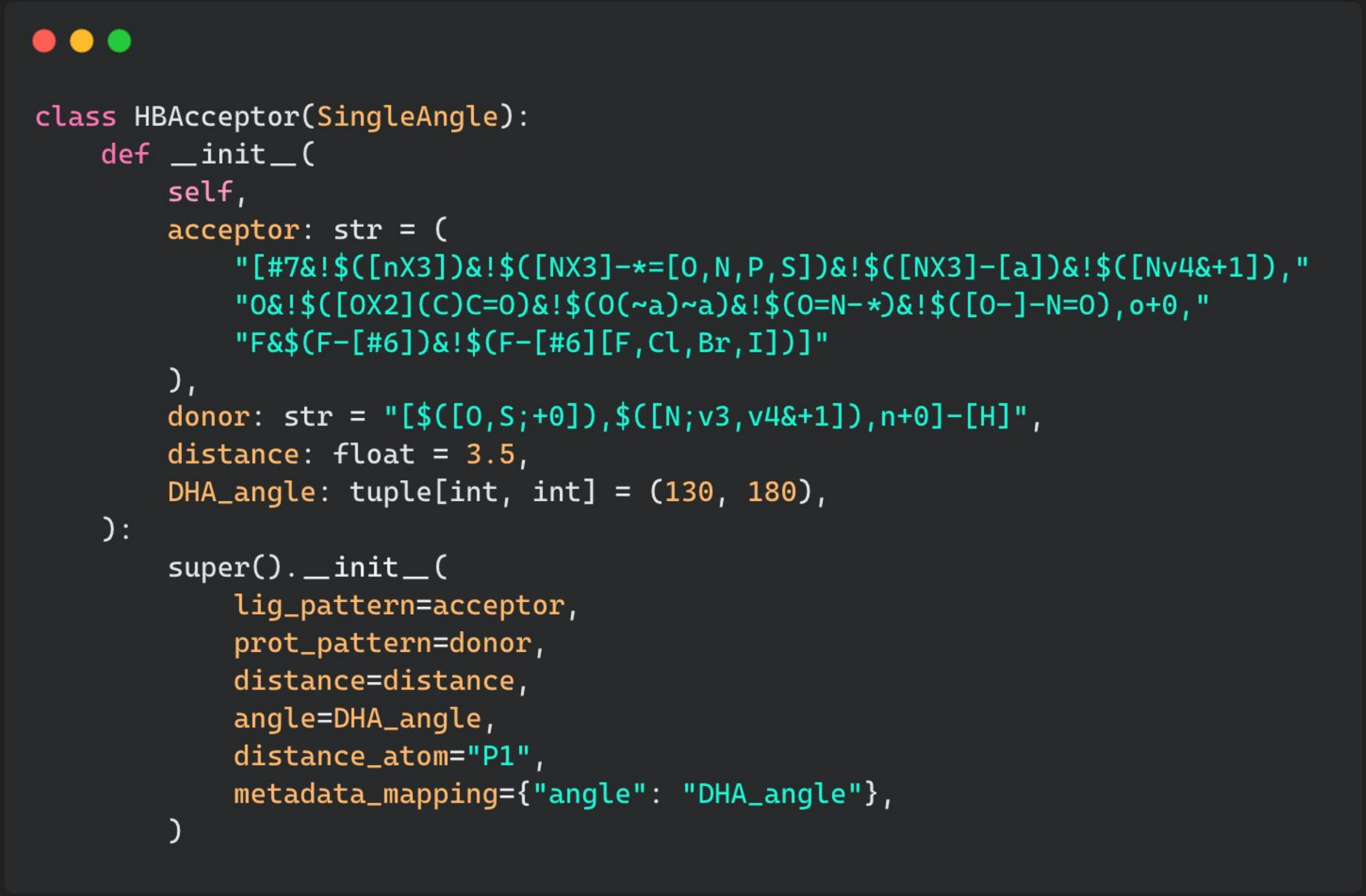


Example: defining a new interaction



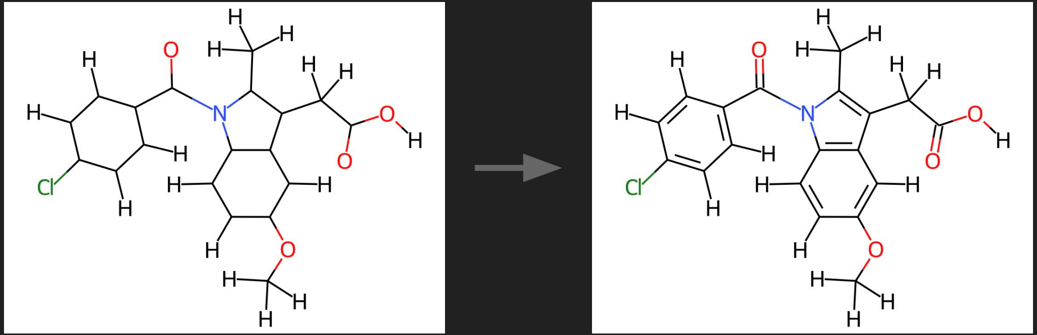
```python
class ExampleInteraction(Interaction):
    def __init__(
        self,
        smarts_acceptor: str = "[#7,#8]",
        smarts_donor: str = "[#7,#8]-[H]",
        distance_threshold: float = 3.5,
    ) → None:
        """
        Validate, transform and store the required parameters.
        """

    def detect(
        self, ligand_residue: Residue, protein_residue: Residue
    ) → Iterator[dict]:
        """Implementation of the interaction detection."""
        ...
        if distance ≤ self.distance_threshold:
            yield self.metadata(
                ligand_residue,
                protein_residue,
                ligand_atom_indices,
                protein_atom_indices,
                distance=distance,
            )
```

Example: HBAcceptor



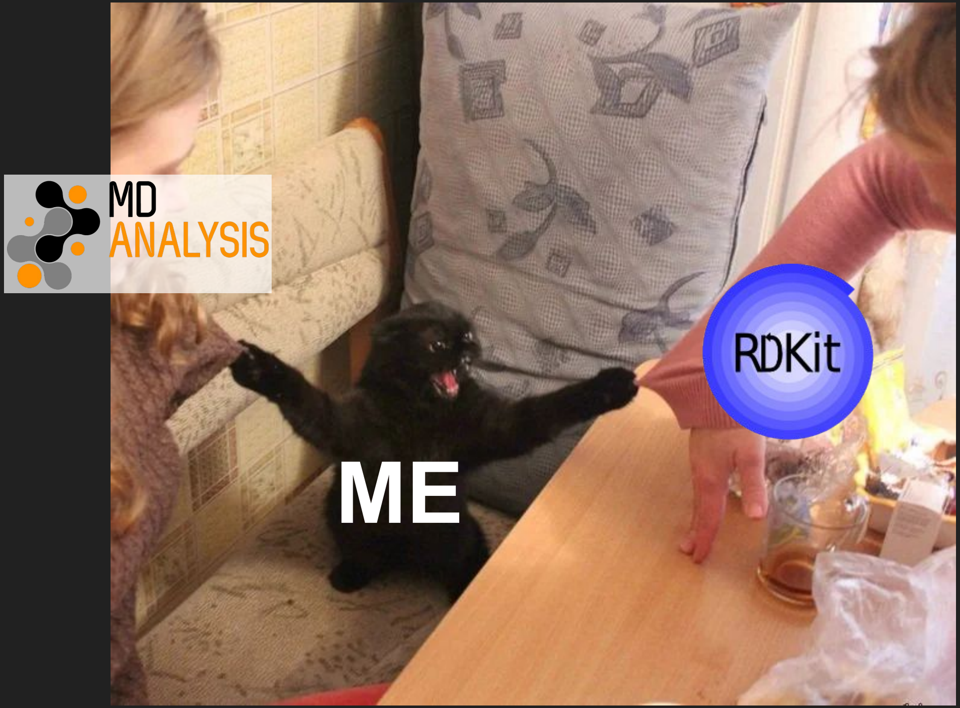
```python
class HBAcceptor(SingleAngle):
    def __init__(
        self,
        acceptor: str = (
            "[#7&!$([nX3])&!$([NX3]-*=[O,N,P,S])&!$([NX3]-[a])&!$([Nv4&+1]),"
            "O&!$([OX2](C)C=O)&!$(O(~a)~a)&!$(O=N-*)&!$([O-]-N=O),o+0,"
            "F&$(F-[#6])&!$(F-[#6][F,Cl,Br,I])]"
        ),
        donor: str = "[$([O,S;+0]),$([N;v3,v4&+1]),n+0]-[H]",
        distance: float = 3.5,
        DHA_angle: tuple[int, int] = (130, 180),
    ):
        super().__init__(
            lig_pattern=acceptor,
            prot_pattern=donor,
            distance=distance,
            angle=DHA_angle,
            distance_atom="P1",
            metadata_mapping={"angle": "DHA_angle"},
        )
```

Interoperability between MDAnalysis and RDKit



Summary

Google
Summer of Code



Motivation

	Knows chemistry	Reads MD trajectories
RDKit	✓	✗
MDAnalysis	✗	✓

From the Universe to RDKit

MD ANALYSIS → RDKit

Requirements

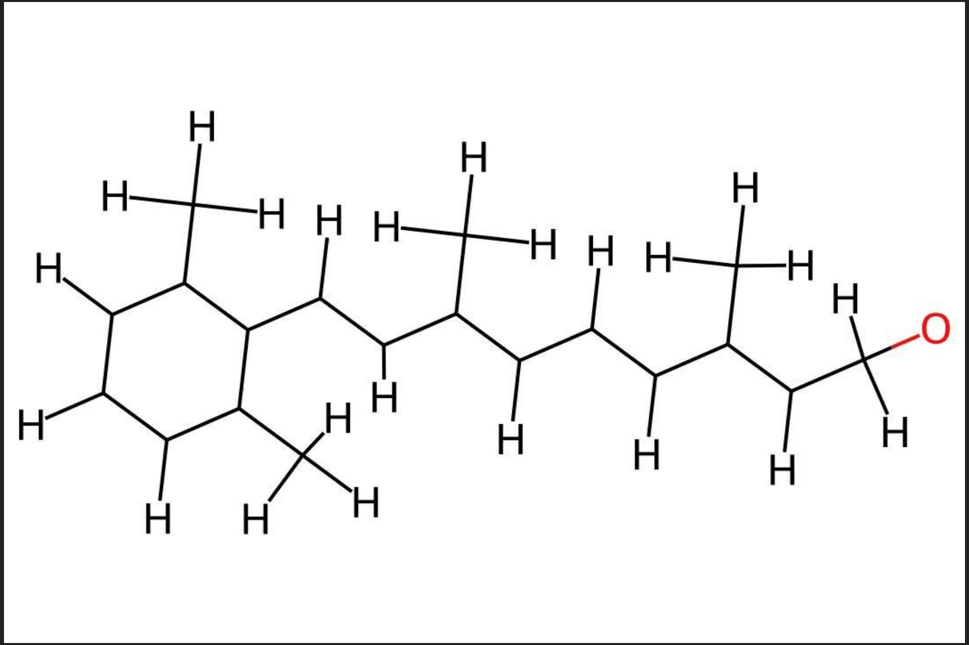
- Elements*
- Bonds*
- **Explicit hydrogens**

Infers bond orders and formal charges from there

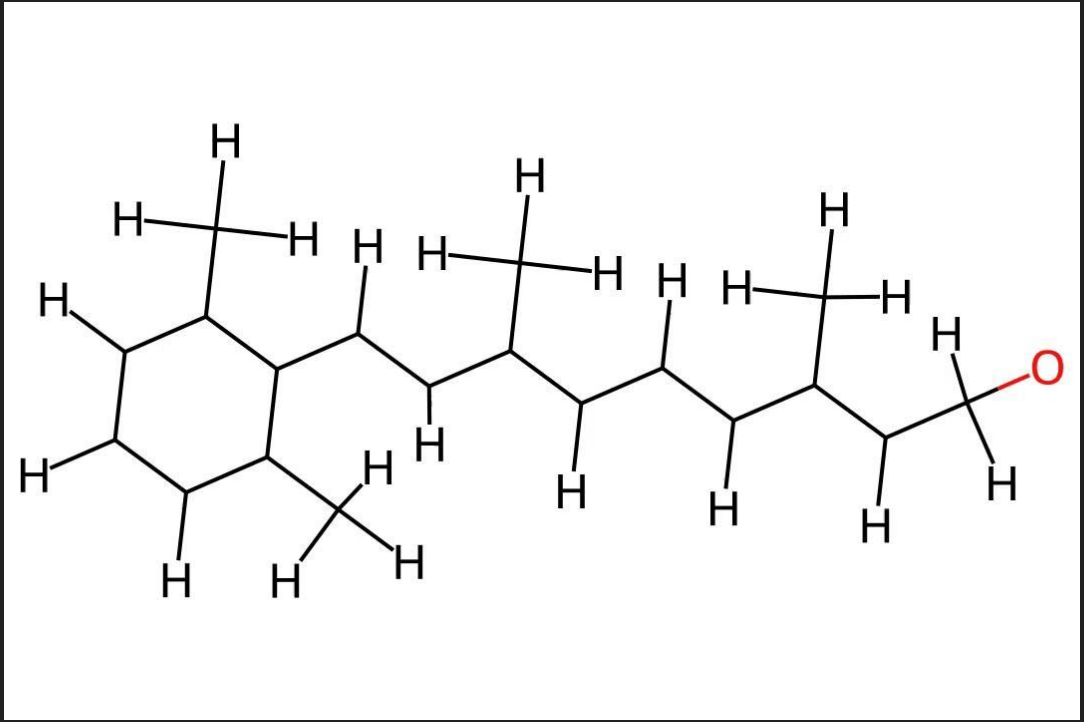
* can be guessed by MDAnalysis

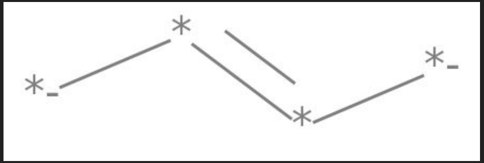
Inferring bond orders and charges

```
For each atom:
    Compare expected vs current valence
    If all valence electrons are paired:
        Continue to next atom
    Else if atom exceeds expected valence:
        Assign (+) charge
    Else:
        If a neighbor can accept a double
        or triple bond:
            Increase bond order
        Else:
            Assign (-) charge
```



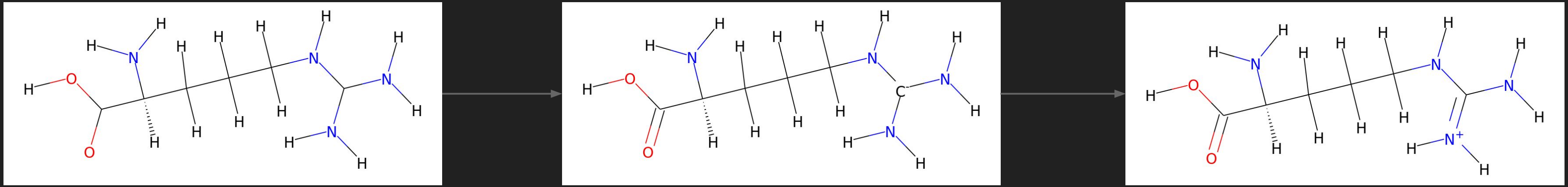
Order dependency of the algorithm

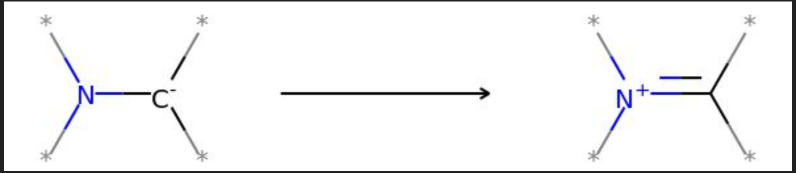




Logical pitfalls

```
For each atom:
      Compare expected vs current valence
      If all valence electrons are paired:
            Continue to next atom
```



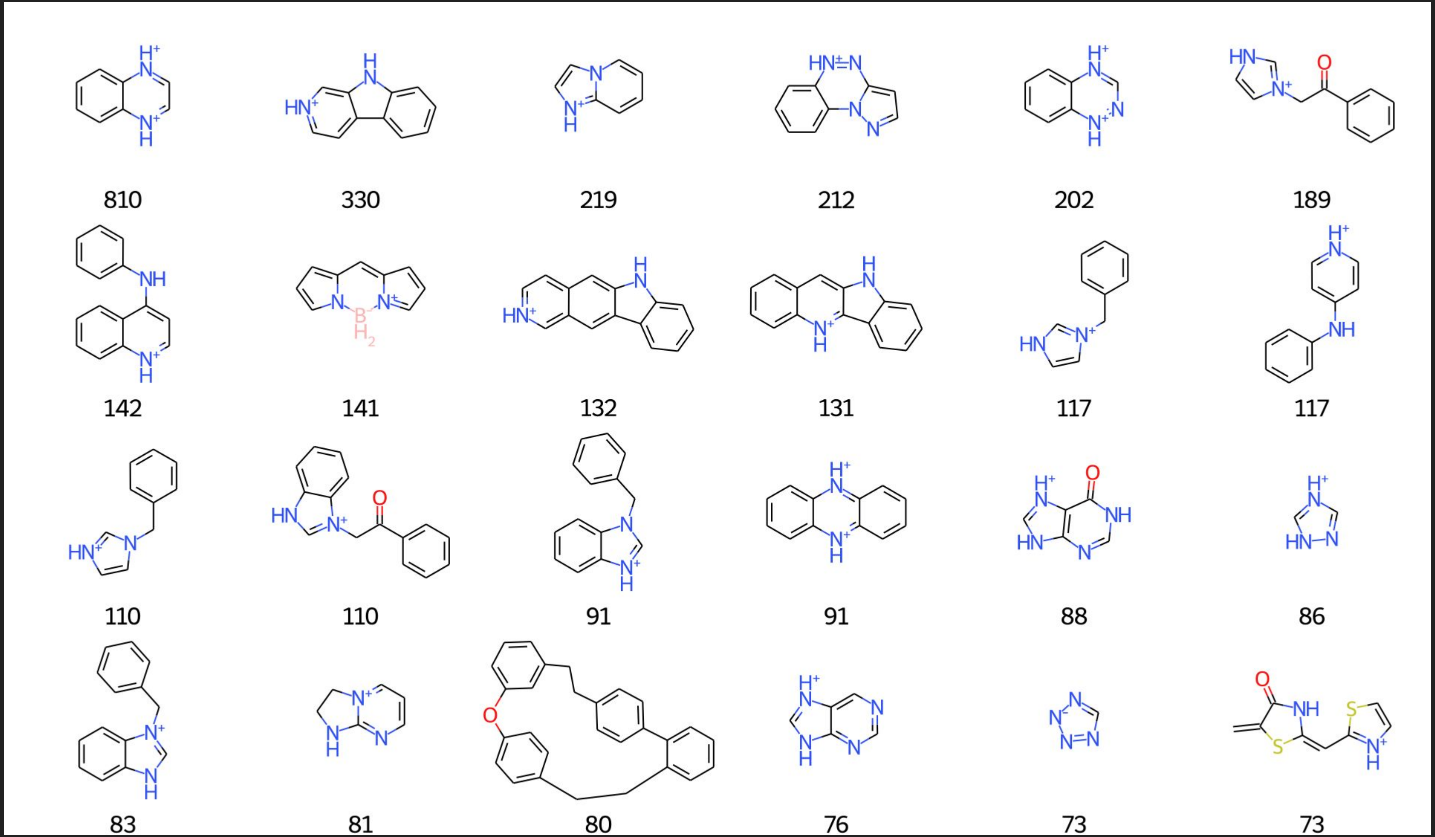


Benchmark

- ChEMBL 33
- 2.2M unique molecules
- Add hydrogens and remove bond orders and charges
- Apply bond order inferring algorithm
- Check whether we match the original molecule / a resonance form
- 99.2% accurate

github.com/MDAnalysis/RDKitConverter-benchmark

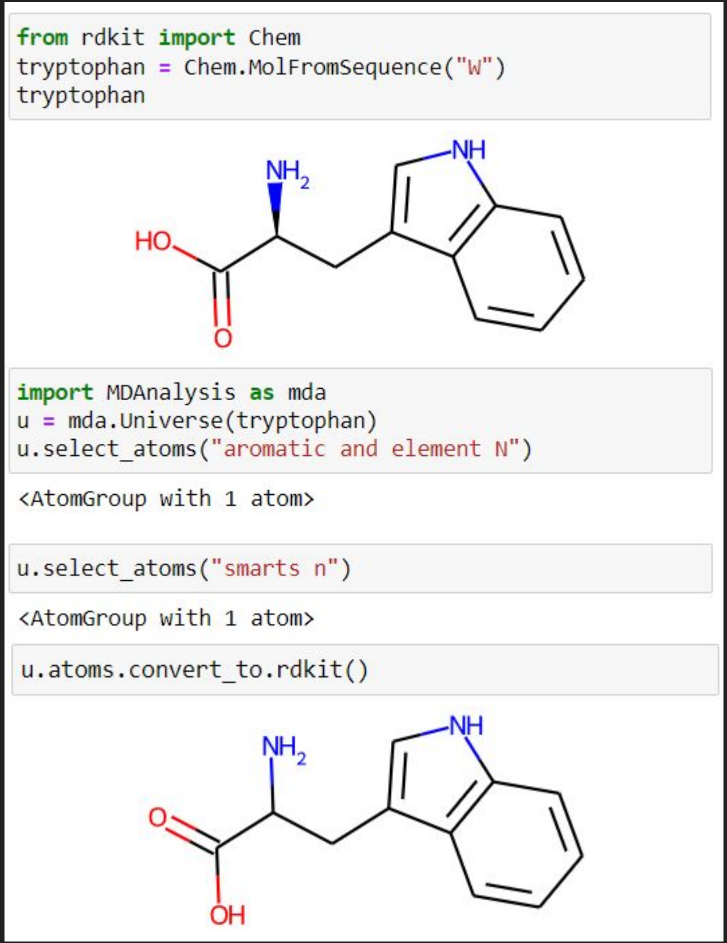
Common failing scaffolds



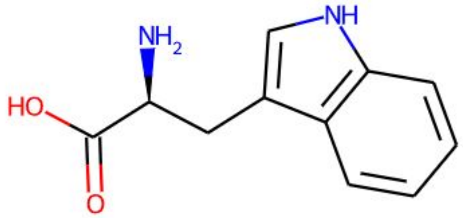
[Scott O. B. and Chan A. W. E. (2020) Bioinformatics]

Interoperability features

- Reading from an RDKit molecule
- Converting to an RDKit molecule
- Selection commands
    - aromaticity
    - SMARTS query



```
from rdkit import Chem
tryptophan = Chem.MolFromSequence("W")
tryptophan
```



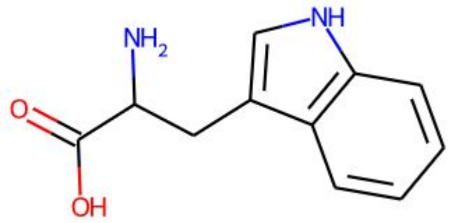
```
import MDAnalysis as mda
u = mda.Universe(tryptophan)
u.select_atoms("aromatic and element N")
```

```
<AtomGroup with 1 atom>
```

```
u.select_atoms("smarts n")
```

```
<AtomGroup with 1 atom>
```

```
u.atoms.convert_to.rdkit()
```



Live demo!

Conclusion

- Can generate IFPs from MD trajectories and more
- Works with any combination of ligand, protein, DNA or RNA molecules
- Configurable and extensible
- Integrates nicely with the typical Python data-science/cheminformatics stack
- Includes many visualisation options

prolif.readthedocs.io

github.com/chemosim-lab/ProLIF

Future work

ProLIF

- CLI
- More plots
- Rewrite parts of the code in Cython for performance
- More interaction types
- More fingerprint types

MDAnalysis RDKitConverter

- Add RDKit's own code for bond order inferring (rdDetermineBonds)
- Add RDKit's template matching AssignBondOrdersFromTemplate
- Fix problems revealed by the benchmark

Acknowledgments

PhD Supervisor

- Sébastien Fiorucci

ProLIF contributors

- Code: Pavel Polishchuk, René Hamburger
- Conda packaging: Hadrien Mary
- Docs: Irfan Alibay, Muhammad Radifar

MDAnalysis

- GSoC mentors: Richard Gowers, Irfan Alibay, Fiona Naughton
- RDKitConverter benchmark: Oliver Beckstein, Irfan Alibay
- MDAKits: Irfan Alibay