

ProLint2

Analyzing lipid-protein interactions in MD simulations

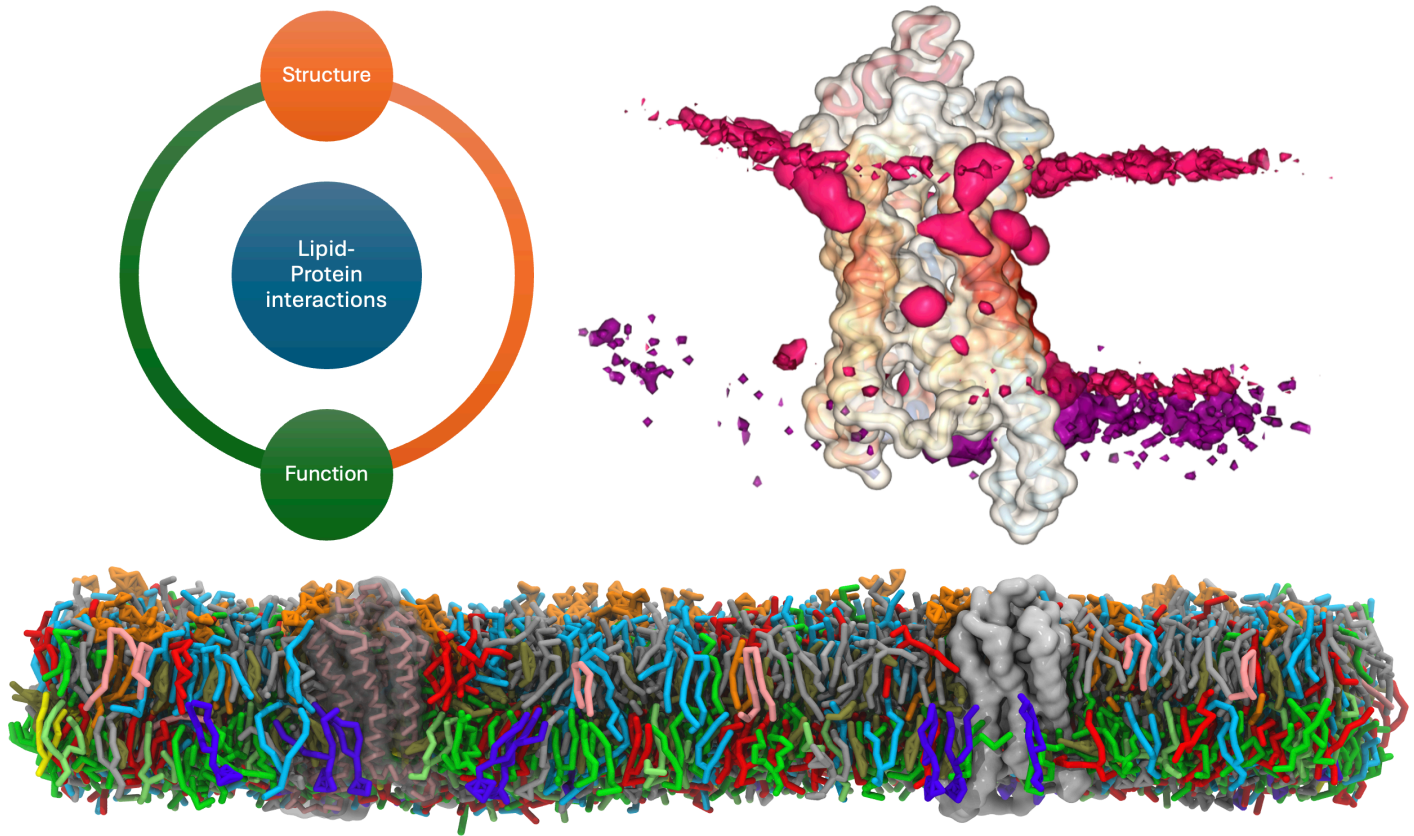
Daniel P. Ramirez-Echemendia

(Martini Workshop - August 12th, 2025)

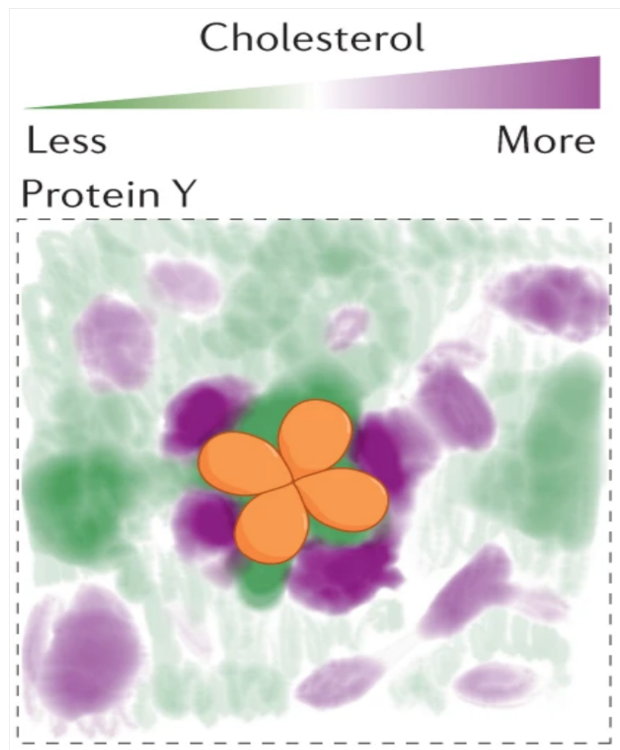




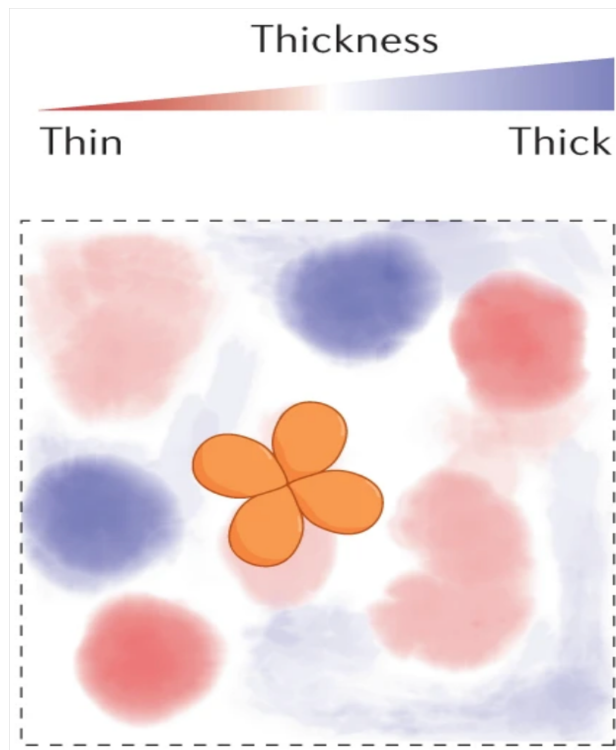
Why do we study lipid-protein interactions?



Why do we study lipid-protein interactions?

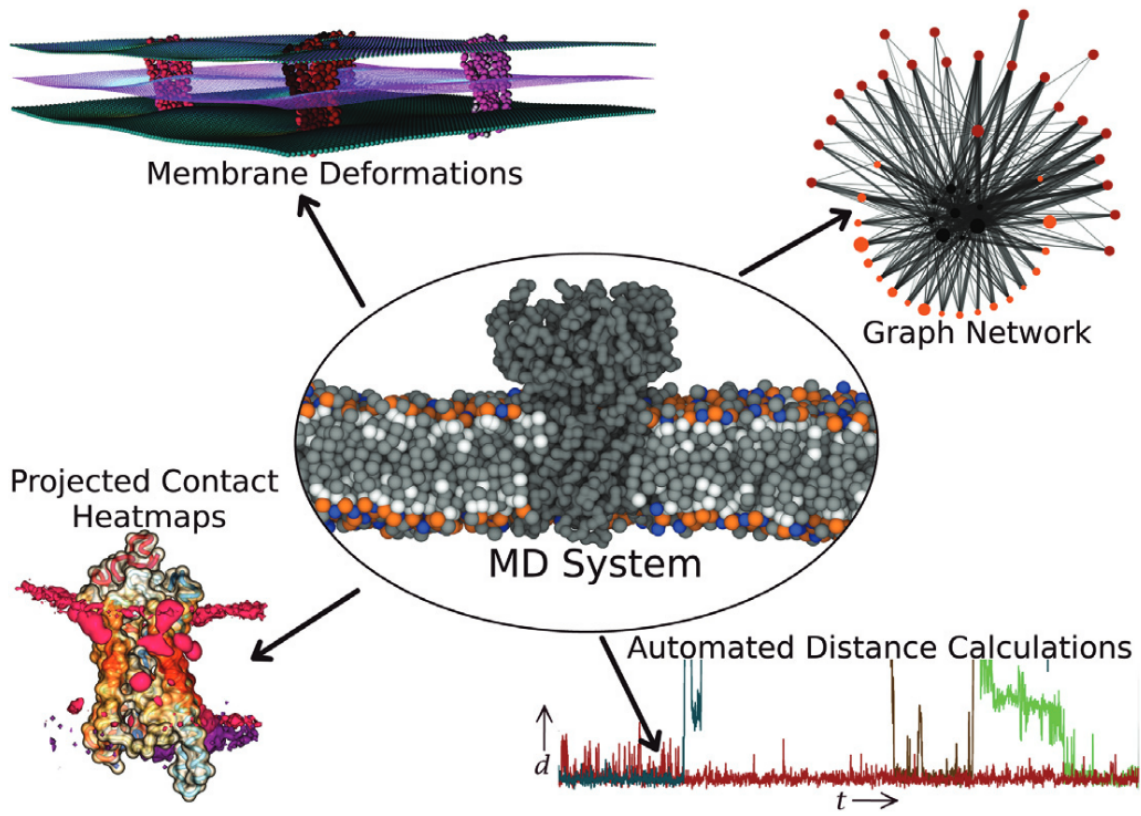


Allosteric-like effect

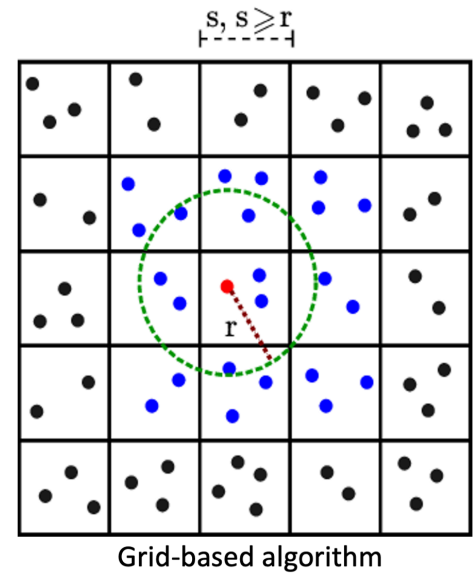
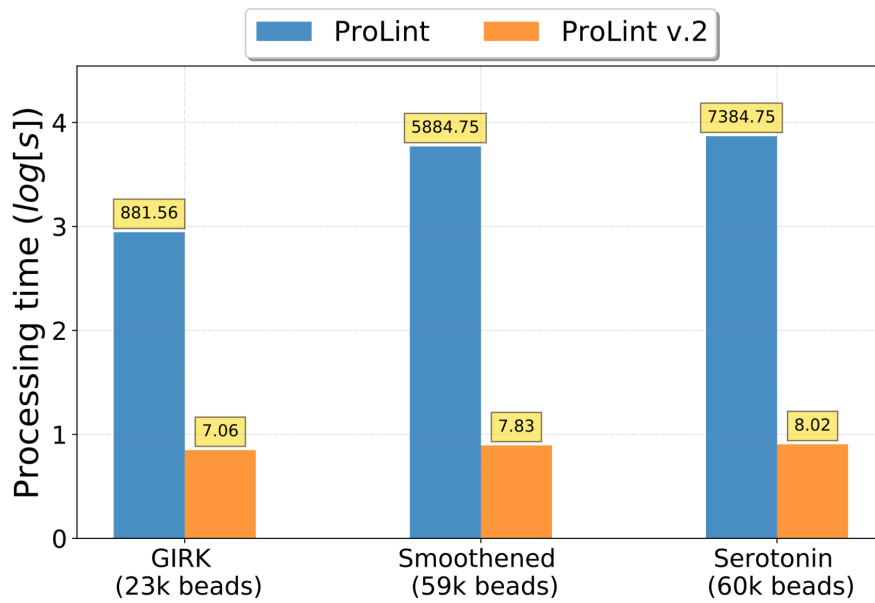


Solvent-like effect

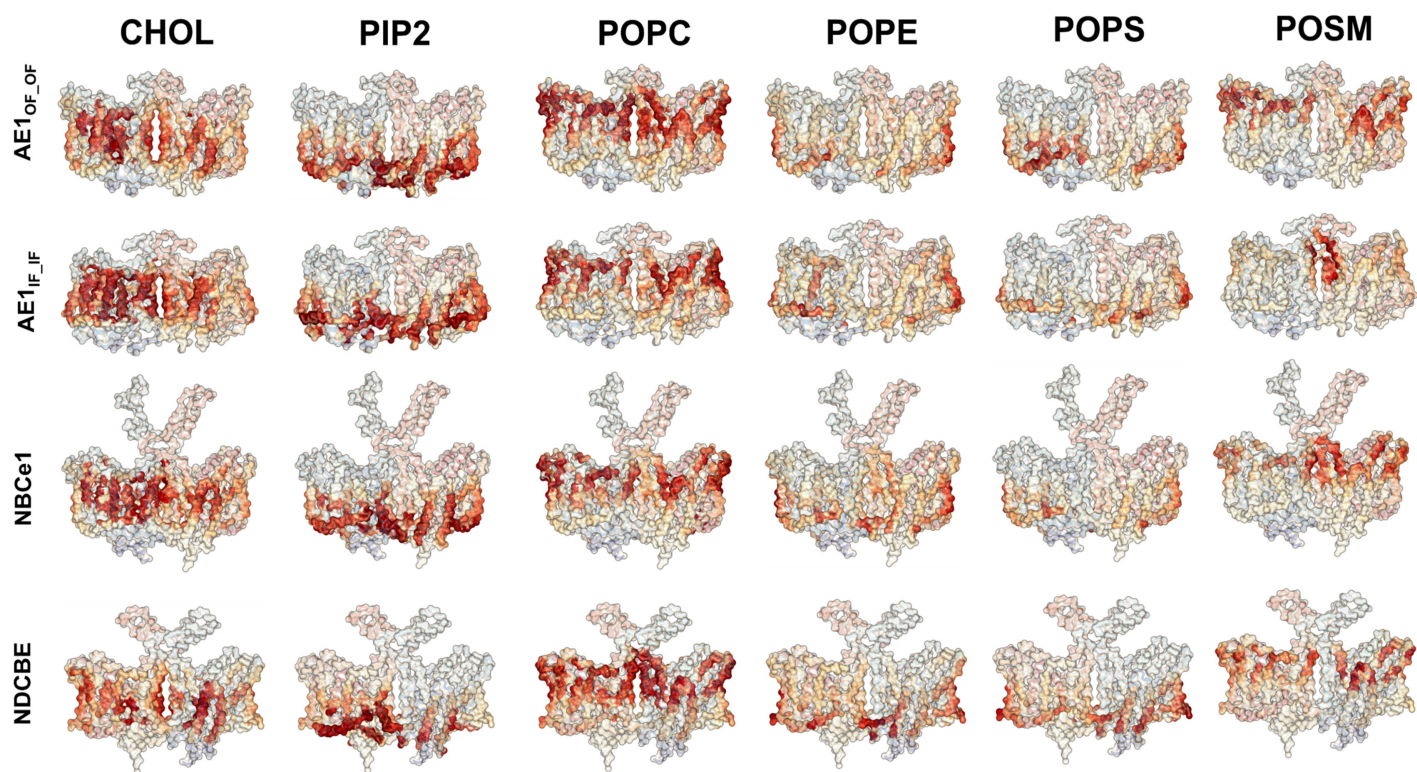
ProLint and ProLint2



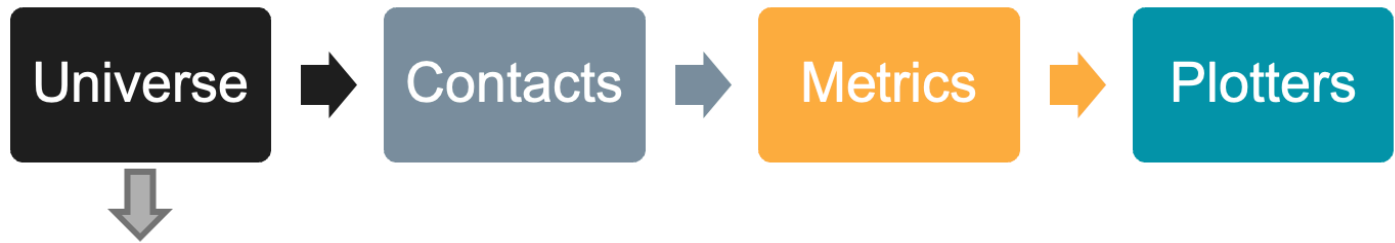
ProLint2 Benchmarks



Martini and ProLint2 in action

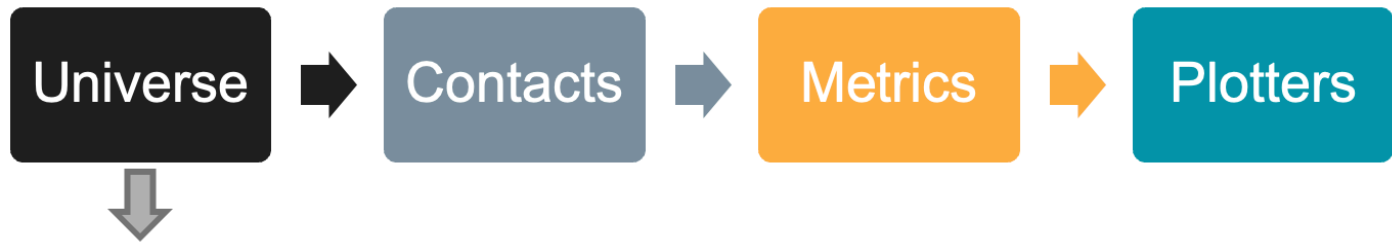


ProLint2 design (sequential flow)



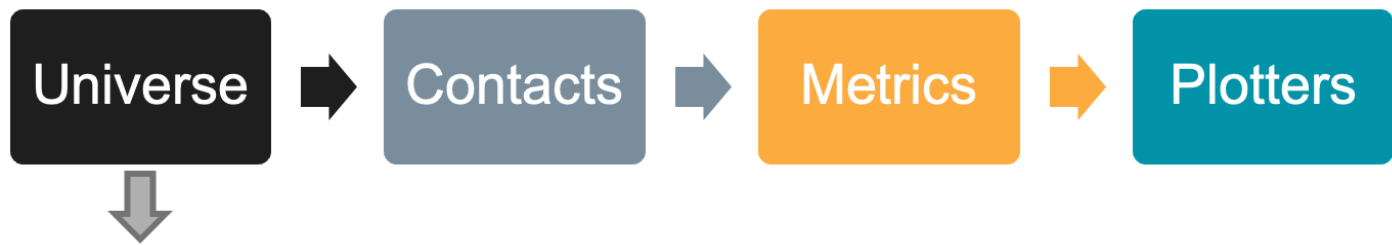
```
1 # Import the ProLint2 Universe
2 from prolint2 import Universe
```

ProLint2 design (sequential flow)



```
1 # Import the ProLint2 Universe
2 from prolint2 import Universe
3
4 # Gather the trajectory data
5 from prolint2.sampledata import GIRKDataSample
6 GIRK = GIRKDataSample()
```

ProLint2 design (sequential flow)



```
1 # Import the ProLint2 Universe
2 from prolint2 import Universe
3
4 # Gather the trajectory data
5 from prolint2.sampledata import GIRKDataSample
6 GIRK = GIRKDataSample()
7
8 # Initialize the Universe
9 u = Universe(GIRK.coordinates, GIRK.trajectory)
```

ProLint2 design (sequential flow)



```
1 u.atoms.n_atoms
```

23820

```
1 u.residues.n_residues
```

3240

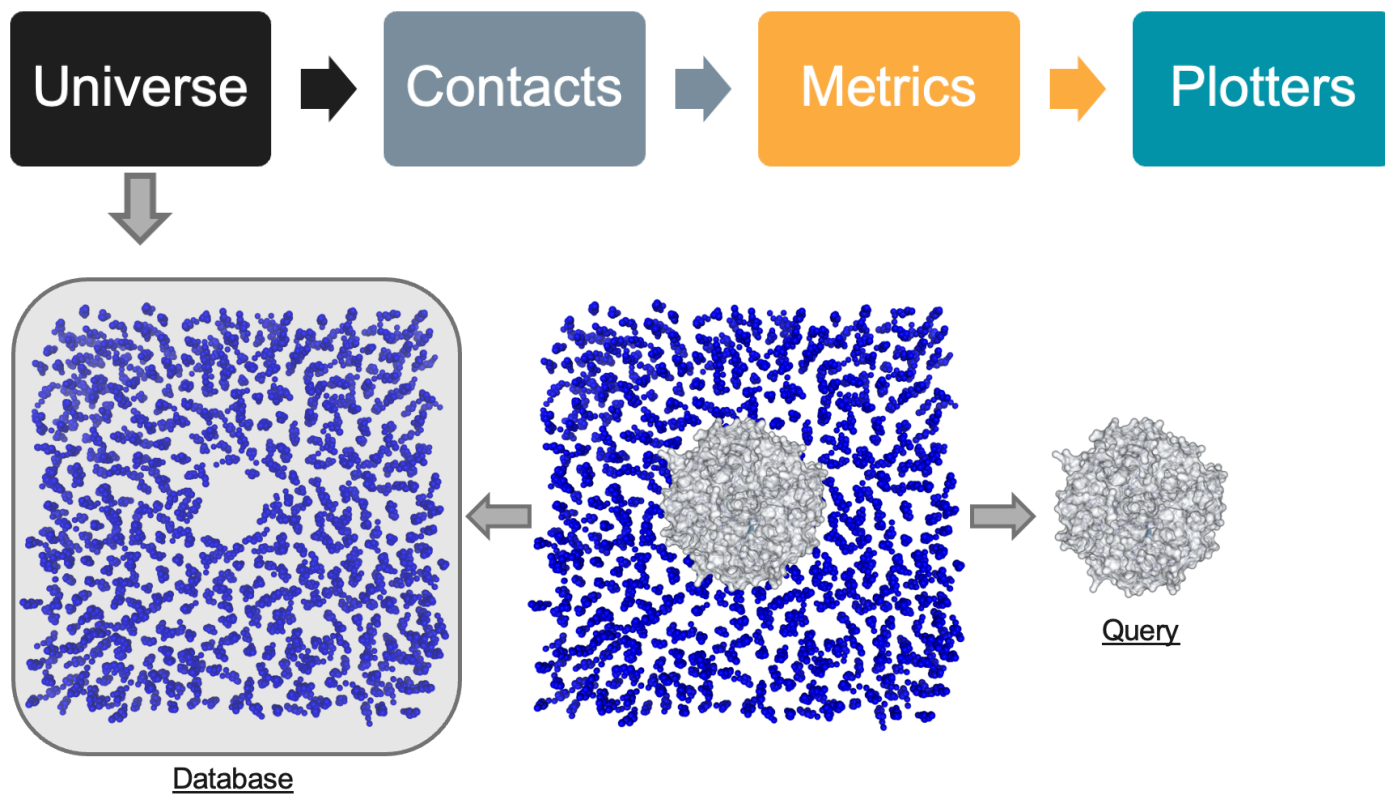
```
1 u.residues.resnames
```

array(['ARG', 'GLN', 'ARG', ..., 'CHOL', 'CHOL', 'CHOL'], dtype=object)

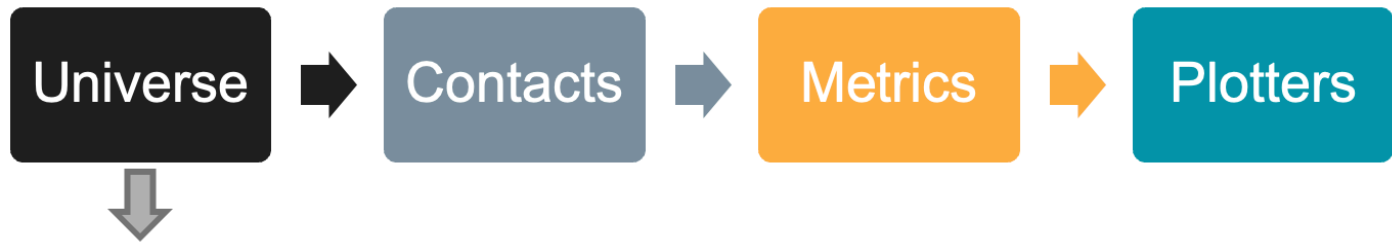
```
1 u.trajectory.n_frames
```

1251

ProLint2 design (sequential flow)



ProLint2 design (sequential flow)



Extracting database information

```
1 u.database.residues.n_residues
```

```
1956
```

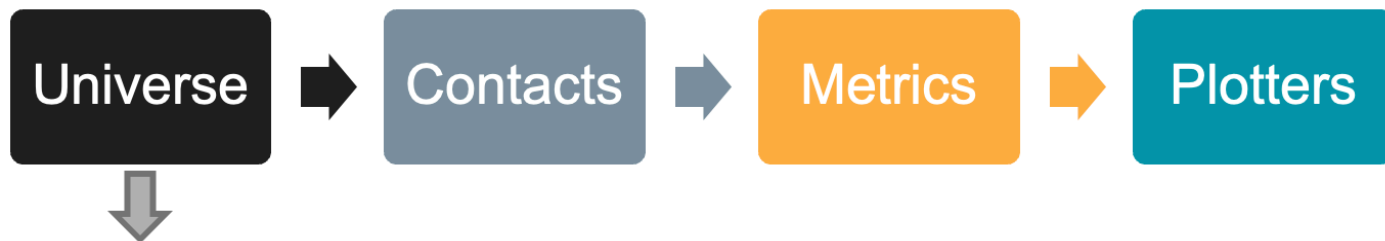
```
1 u.database.resname_counts
```

```
Counter({'POPE': 652, 'POPS': 652, 'CHOL': 652})
```

```
1 u.database.unique_resnames
```

```
array(['CHOL', 'POPE', 'POPS'], dtype=object)
```

ProLint2 design (sequential flow)



```
1 u.query.residues.n_residues
```

```
1284
```

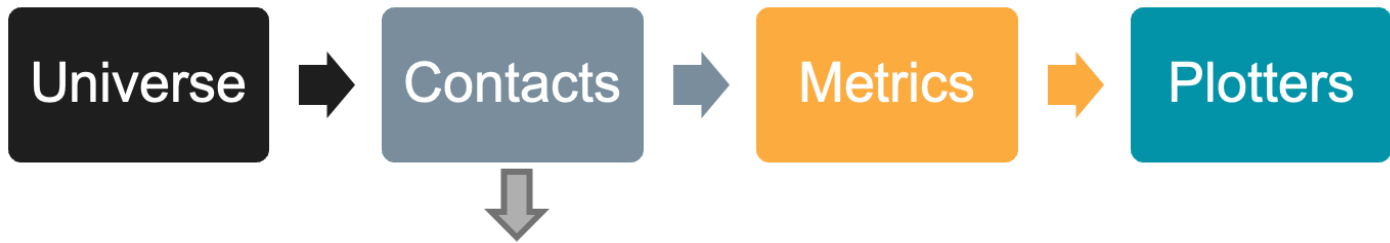
```
1 u.query.filter_resids_by_resname(resname='CYS')
```

```
array([ 11,  80, 112, 136, 167, 267, 313, 314, 332, 401, 433,
        457, 488, 588, 634, 635, 653, 722, 754, 778, 809, 909,
        955, 956, 974, 1043, 1075, 1099, 1130, 1230, 1276, 1277])
```

```
1 u.query.resname_counts['HIS']
```

```
28
```

ProLint2 design (sequential flow)



```

1 # Initialize a Contacts instance
2 contacts = u.compute_contacts(cutoff=7, backend='Serial')
  
```

100%  1251/1251 [00:02<00:00, 440.35it/s]

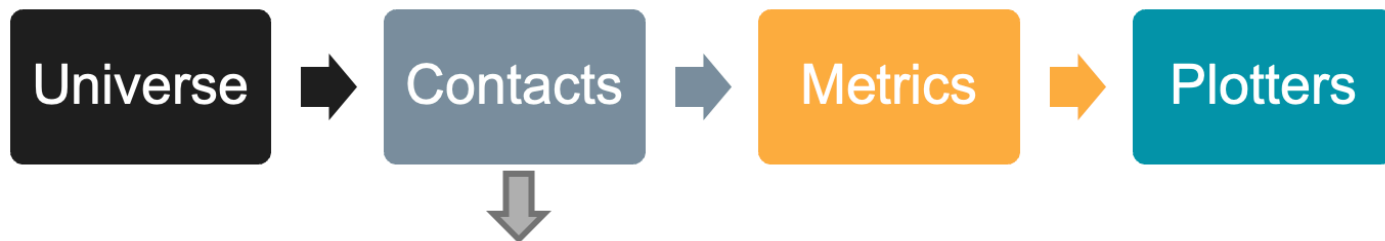
Access the contacts information

```
1 contacts.contact_frames # --> {ResID: {LipID: [frames]}}
```

```
1 contacts.contact_frames[14][2800]
```

```
[2, 7, 8, 13, 18, 20, 21, 22, 24, 27, 33, 57]
```

ProLint2 design (sequential flow)



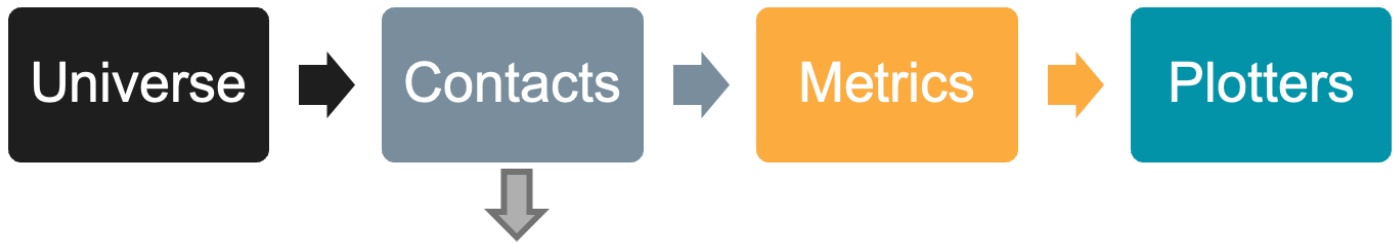
Get lipids by residue ID

```
1 contacts.get_lipids_by_residue_id(residue_id=14)[:10]  
[2267, 2270, 2274, 2299, 2301, 2307, 2308, 2316, 2326, 2347]
```

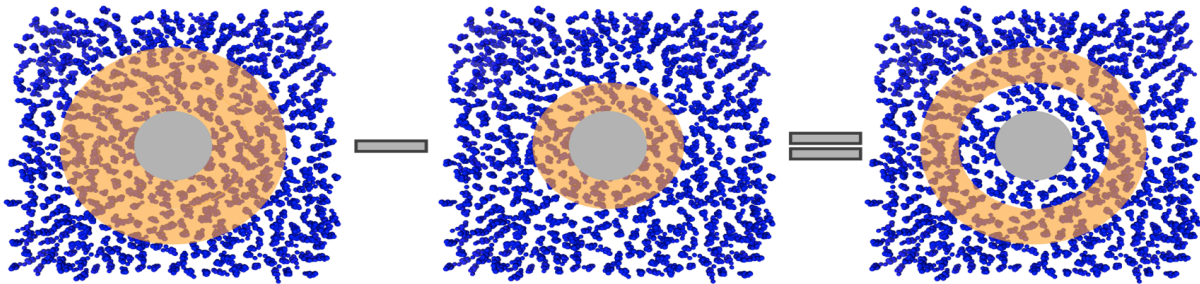
Get residues by lipid ID

```
1 contacts.get_residues_by_lipid_id(lipid_id=2800)[:10]  
[14, 16, 17, 24, 35, 36, 37, 38, 39, 41]
```

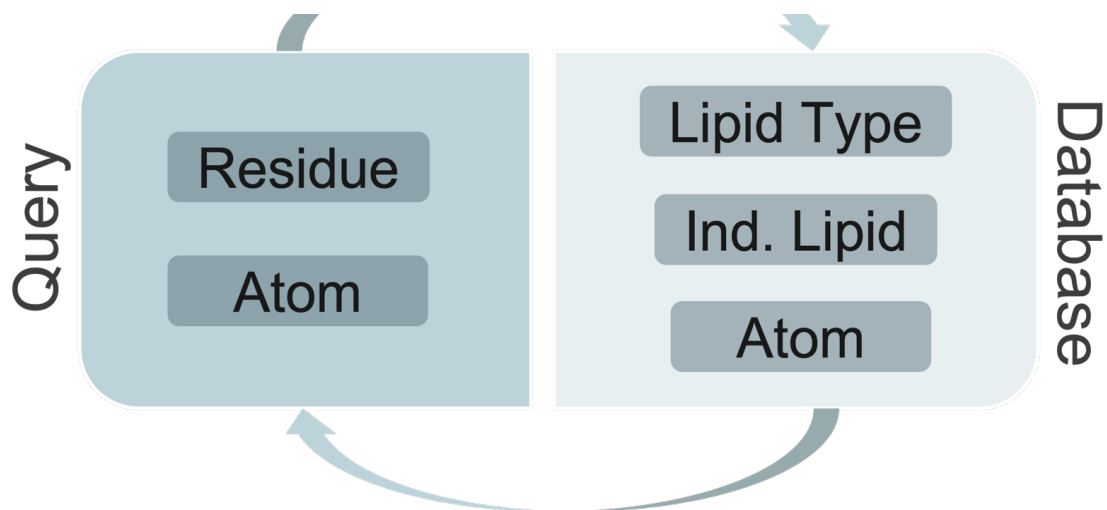
ProLint2 design (sequential flow)



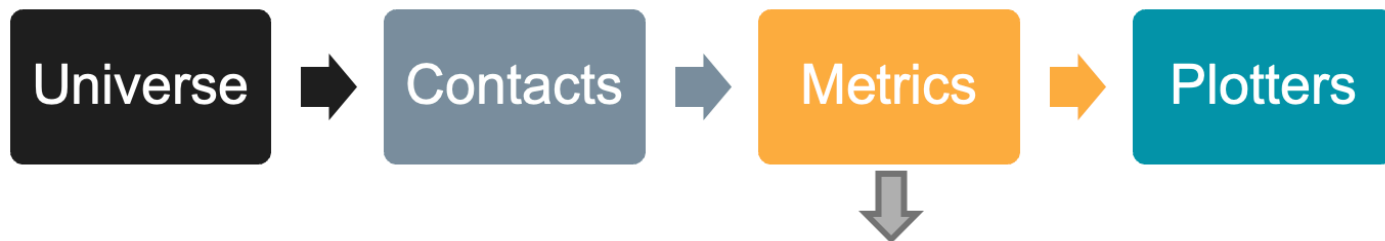
```
1 contacts1 = u.compute_contacts(cutoff=9)
2 contacts2 = u.compute_contacts(cutoff=6)
3 contacts_diff = contacts2.difference(contacts1) # -> contacts1 - contacts2
```



ProLint2 design (sequential flow)

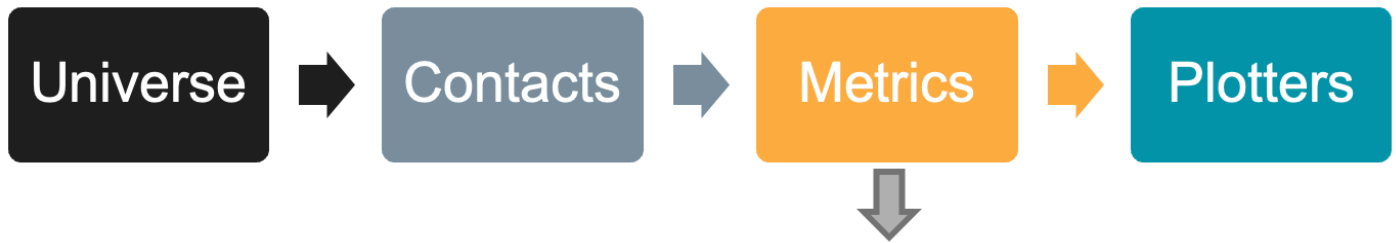


ProLint2 design (sequential flow)



```
1 from prolint2.metrics import Metric, MeanMetric, SumMetric, MaxMetric
2
3 # SumMetric --> Sum of all interaction windows with the same lipid type
4 #               normalized over the entire trajectory.
5 # MaxMetric --> Longest interaction window with the same lipid type
6 #               normalized over the entire trajectory.
7 # MeanMetric --> Mean duration of interaction windows with the same lipid
8 #               type nomalized over the entire trajectory.
9 #               SumMetric >= Maxmetric >= MeanMetric
```

ProLint2 design (sequential flow)



```
1 metrics_instance = Metric(contacts, [SumMetric(), MaxMetric(), MeanMetric()
2 metrics = metrics_instance.compute()
3 metrics[1]
```

```
defaultdict(dict,
    {'CHOL': {'SumMetric': 0.0024000000000000002,
    'MaxMetric': 0.0016,
    'MeanMetric': 0.0012000000000000001},
    'POPE': {'SumMetric': 0.0888,
    'MaxMetric': 0.0032,
    'MeanMetric': 0.000896969696969697},
    'POPS': {'SumMetric': 0.2792,
    'MaxMetric': 0.0048000000000000004,
    'MeanMetric': 0.0010496240601503759}})
```

Analysis workflow with ProLint2

1. Identification of relevant lipid types and metrics.

Multiple plots with PointDistributions

```
1 from prolint2.plotting.multiples import MultiplePointDistribution
2
3 MPD = MultiplePointDistribution(u, contacts_out)
4 MPD.create_plot(rows='metrics', decimals=2)
```

Analysis workflow with ProLint2

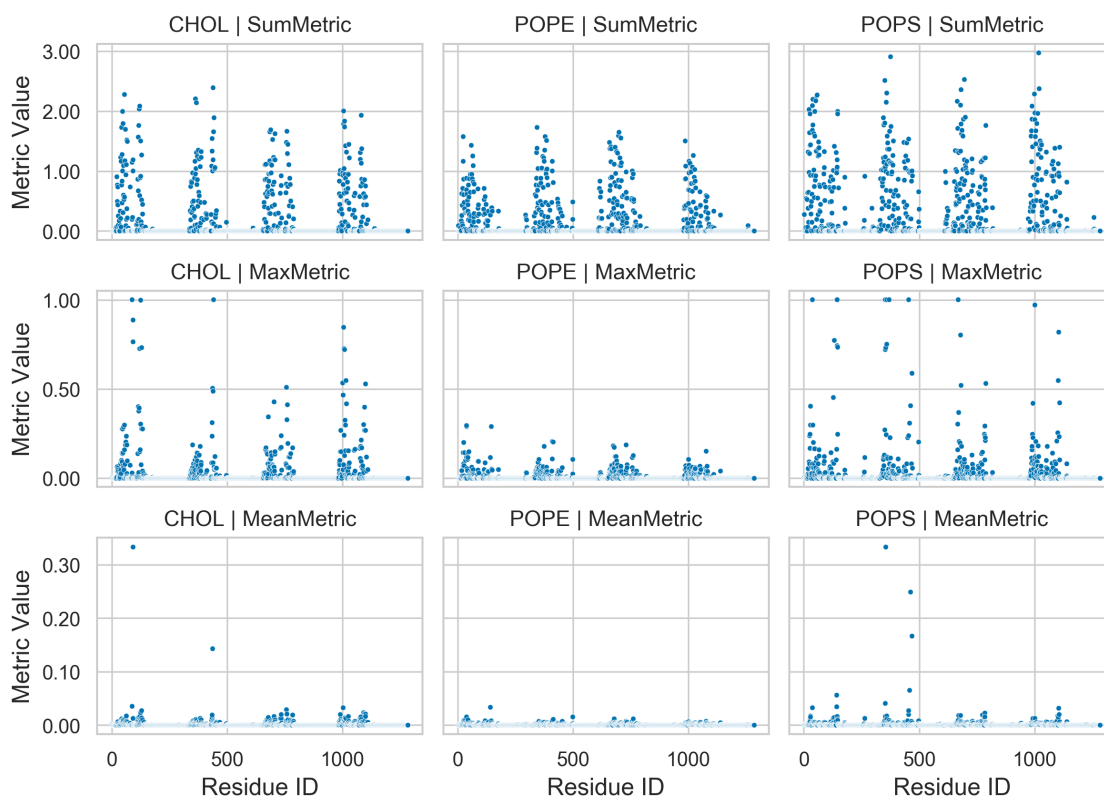
1. Identification of relevant lipid types and metrics.

Multiple plots with PointDistributions

```
1 from prolint2.plotting.multiples import MultiplePointDistribution
2
3 MPD = MultiplePointDistribution(u, contacts_out)
4 MPD.create_plot(rows='metrics', decimals=2)
```

Analysis workflow with ProLint2

Point distributions - Metrics x Lipid types



Analysis workflow with ProLint2

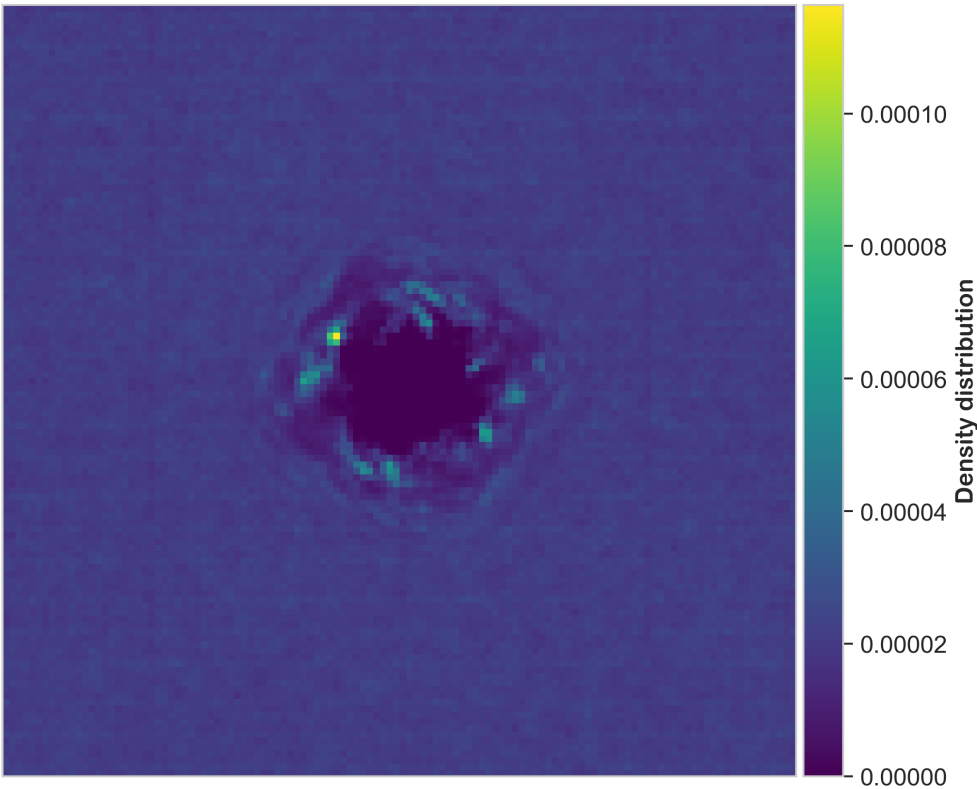
1. Identification of relevant lipid types and metrics.

Density Plots

```
1 from prolint2.plotting import DensityMap
2
3 Dens = DensityMap(u)
4 Dens.create_plot(lipid_type='CHOL',
5                  interpolation='nearest',
6                  cmap='viridis')
```


Analysis workflow with ProLint2

Density map - CHOL



Analysis workflow with ProLint2

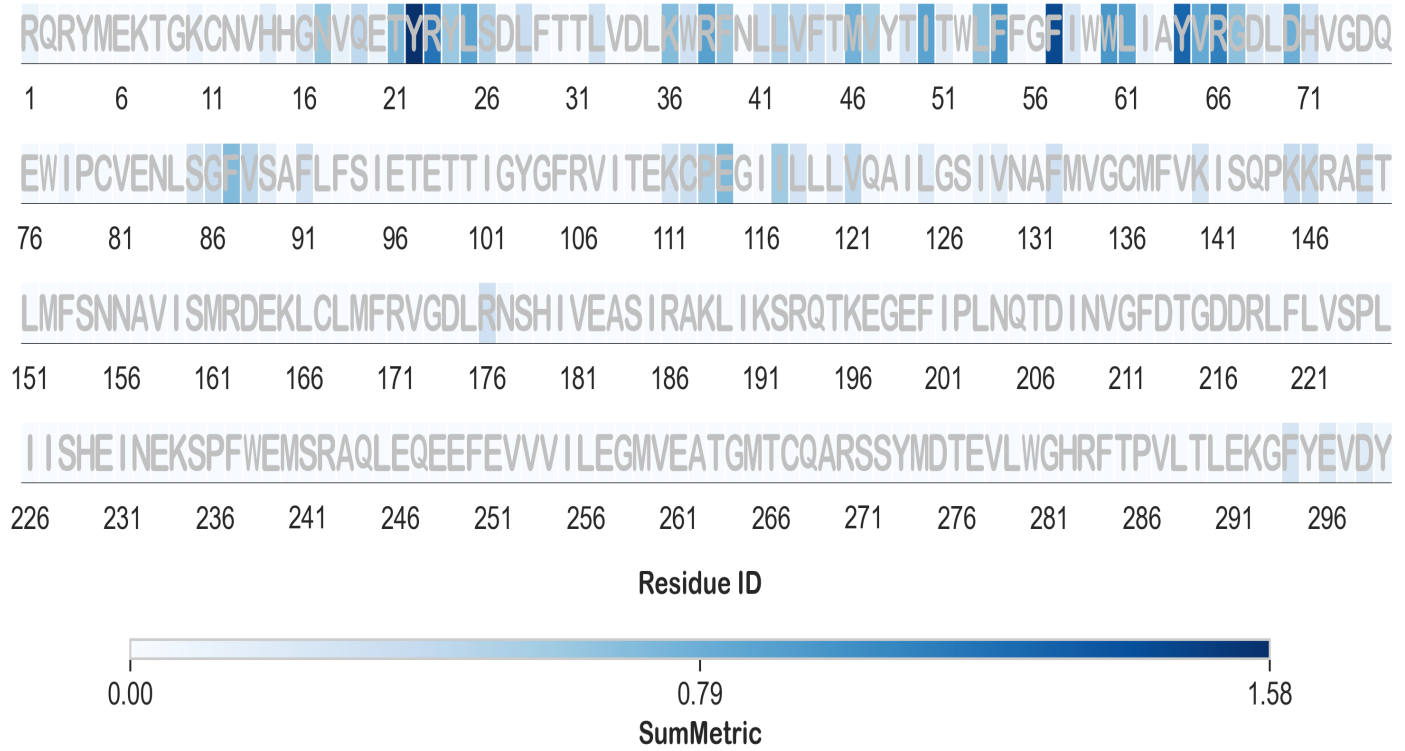
2. Identification of relevant protein residues

Logo plot

```
1 from prolint2.plotting import LogoResidues
2
3 LR = LogoResidues(u, sum_contacts)
4 LR.create_plot( # lipid type and metric name identified in step 1
5     lipid_type='POPE',
6     metric_name='SumMetric',
7     color_logo='Silver',
8     palette='Blues',
9     res_ids=range(1, 300)
10 )
```

Analysis workflow with ProLint2

Logo based on SumMetric



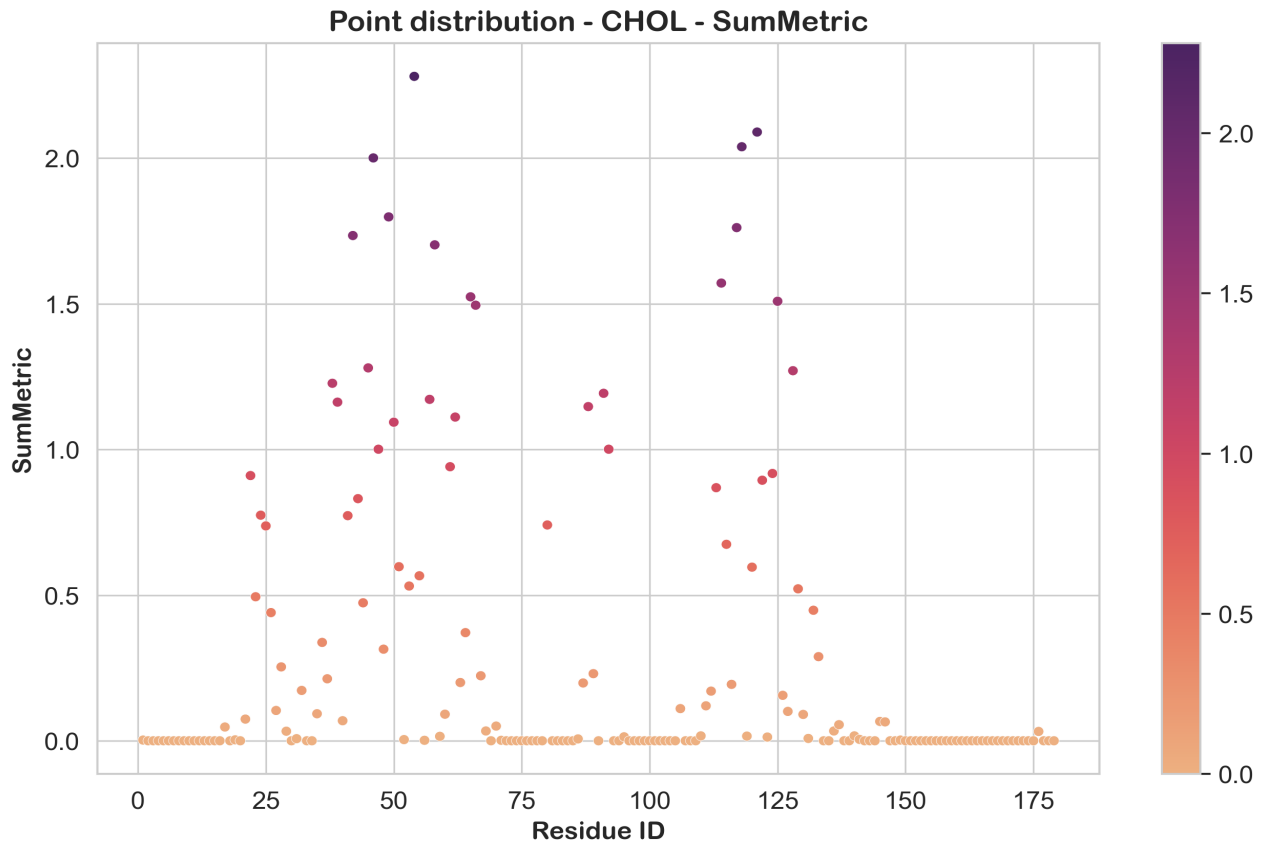
Analysis workflow with ProLint2

2. Identification of relevant protein residues

Single plot with PointDistribution

```
1 from prolint2.plotting import PointDistribution
2
3 PD = PointDistribution(u, sum_contacts)
4 PD.create_plot(
5     lipid_type='CHOL',
6     metric_name='SumMetric',
7     linewidth=0.24,
8     palette='flare',
9     res_ids=range(1, 180)
10 )
```

Analysis workflow with ProLint2



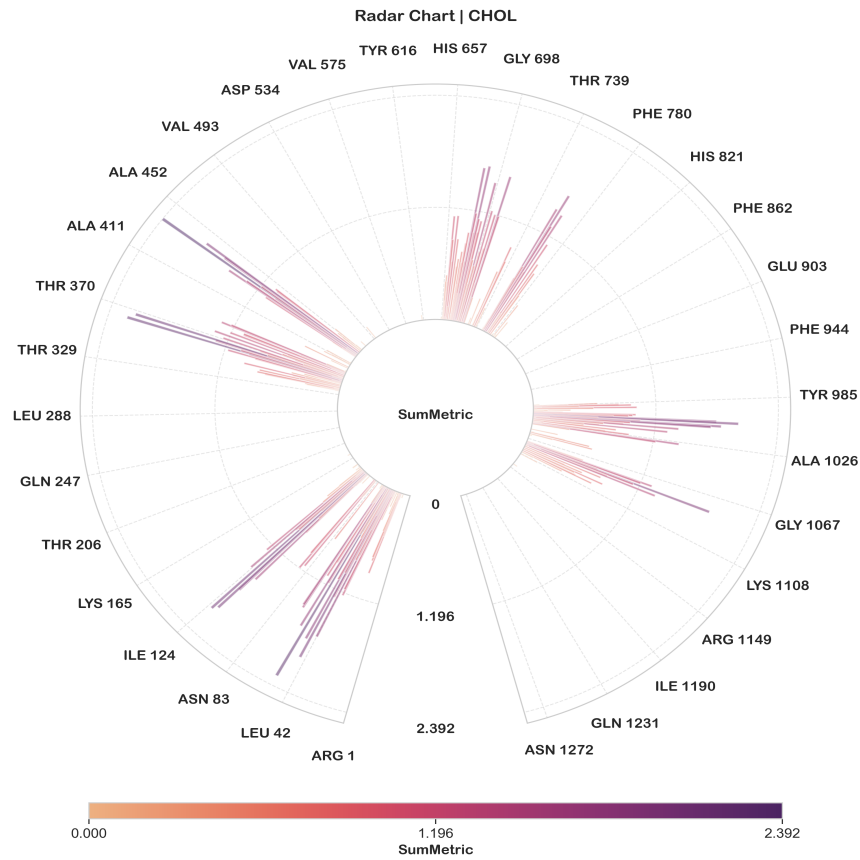
Analysis workflow with ProLint2

2. Identification of relevant protein residues

Single plot with PointDistribution

```
1 from prolint2.plotting import Radar
2
3 RP = Radar(u, sum_contacts)
4 RP.create_plot(
5     lipid='CHOL',
6     metric_name='SumMetric',
7     palette='flare',
8 )
```

Analysis workflow with ProLint2



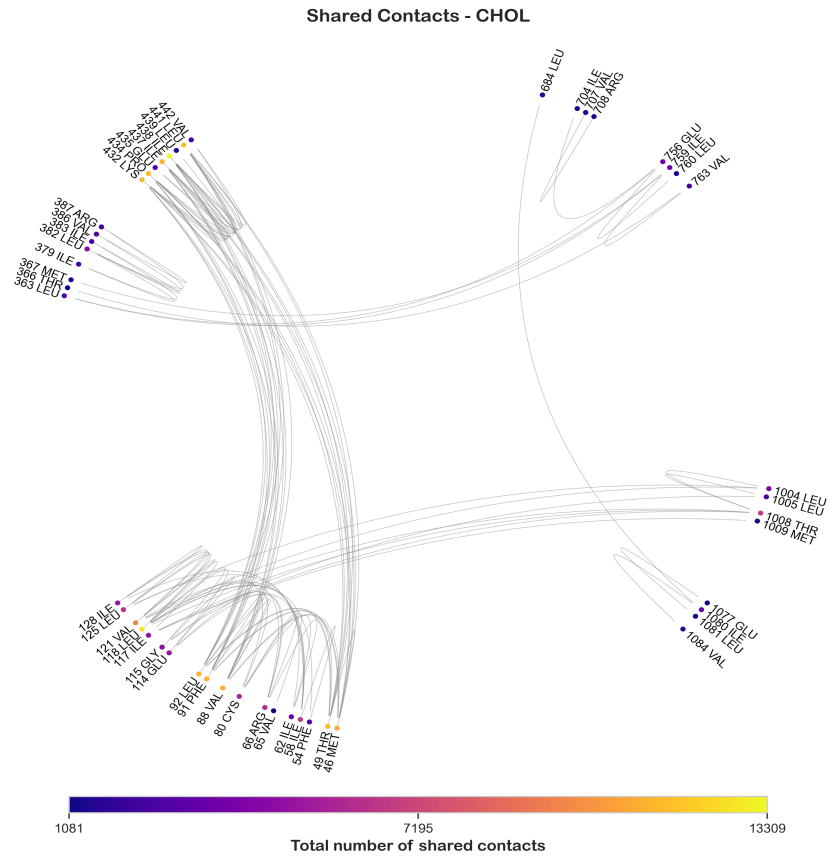
Analysis workflow with ProLint2

2. Identification of relevant protein residues

Shared contacts (Identifying binding sites)

```
1 from prolint2.plotting import SharedContacts
2
3 SCon = SharedContacts(u, contacts)
4 SCon.create_plot(lipid_type='CHOL',
5                  label_size=9,
6                  palette='plasma')
```


Analysis workflow with ProLint2



Analysis workflow with ProLint2

3. Analyzing interactions at the residue level

Identifying individual lipids in relevant interactions

```
1 from prolint2.plotting import DurationGantt
2
3 DG = DurationGantt(u, contacts)
4 DG.get_contact_durations('CHOL', frequency_filter=6).head()
```

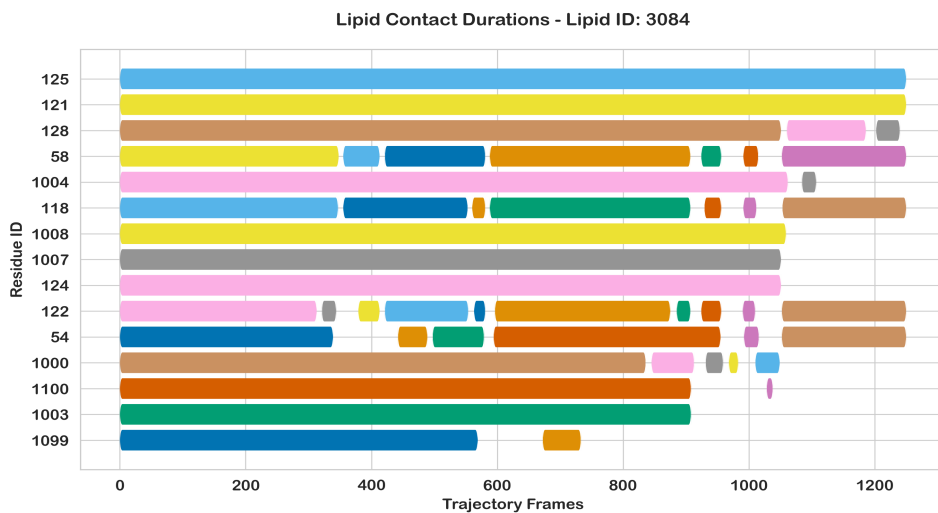
	Lipid ID	Frequency
0	3084	18494.0
1	2951	14591.0
2	3034	13941.0
3	2124	12764.0
4	3185	12459.0

Analysis workflow with ProLint2

3. Analyzing interactions at the residue level

Identifying individual lipids in relevant interactions

```
1 DG.create_plot(lipid_id = 3084, top_filter=15)
```



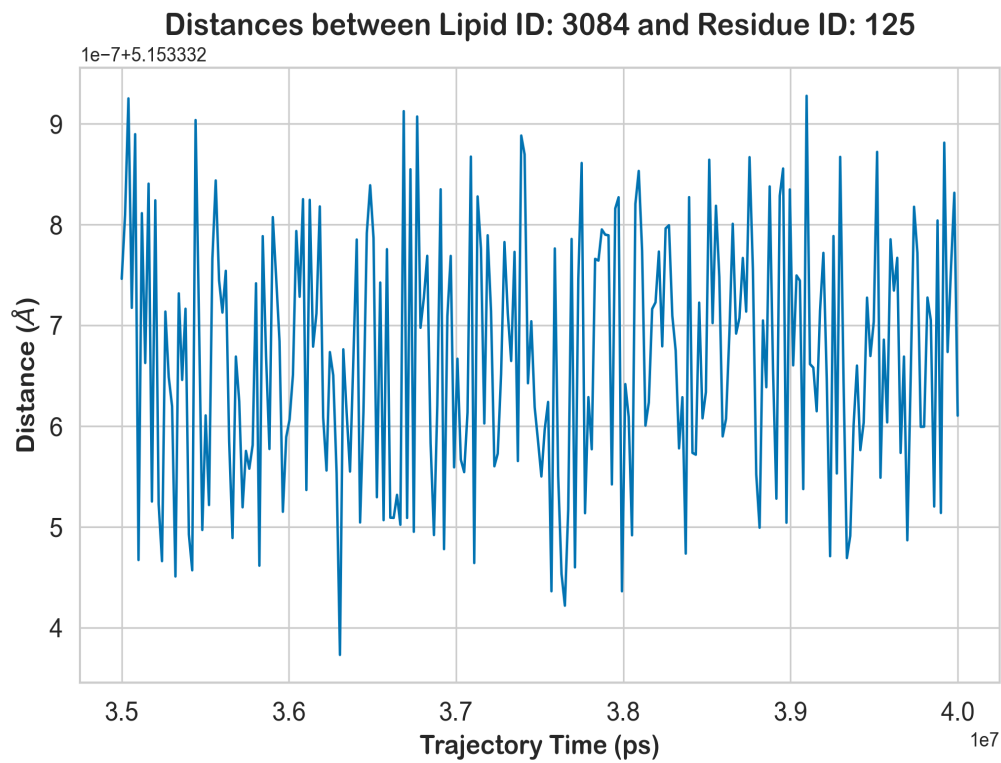
Analysis workflow with ProLint2

3. Analyzing interactions at the residue level

Distance evolution between residues (COM)

```
1 from prolint2.plotting import TwoPointDistanceEvolution
2
3 TPD = TwoPointDistanceEvolution(u)
4 TPD.create_plot(
5     lipid_id = 3084,
6     residue_id = 125,
7     unit = 'time',
8     smooth_line = True)
```

Analysis workflow with ProLint2



Analysis workflow with ProLint2

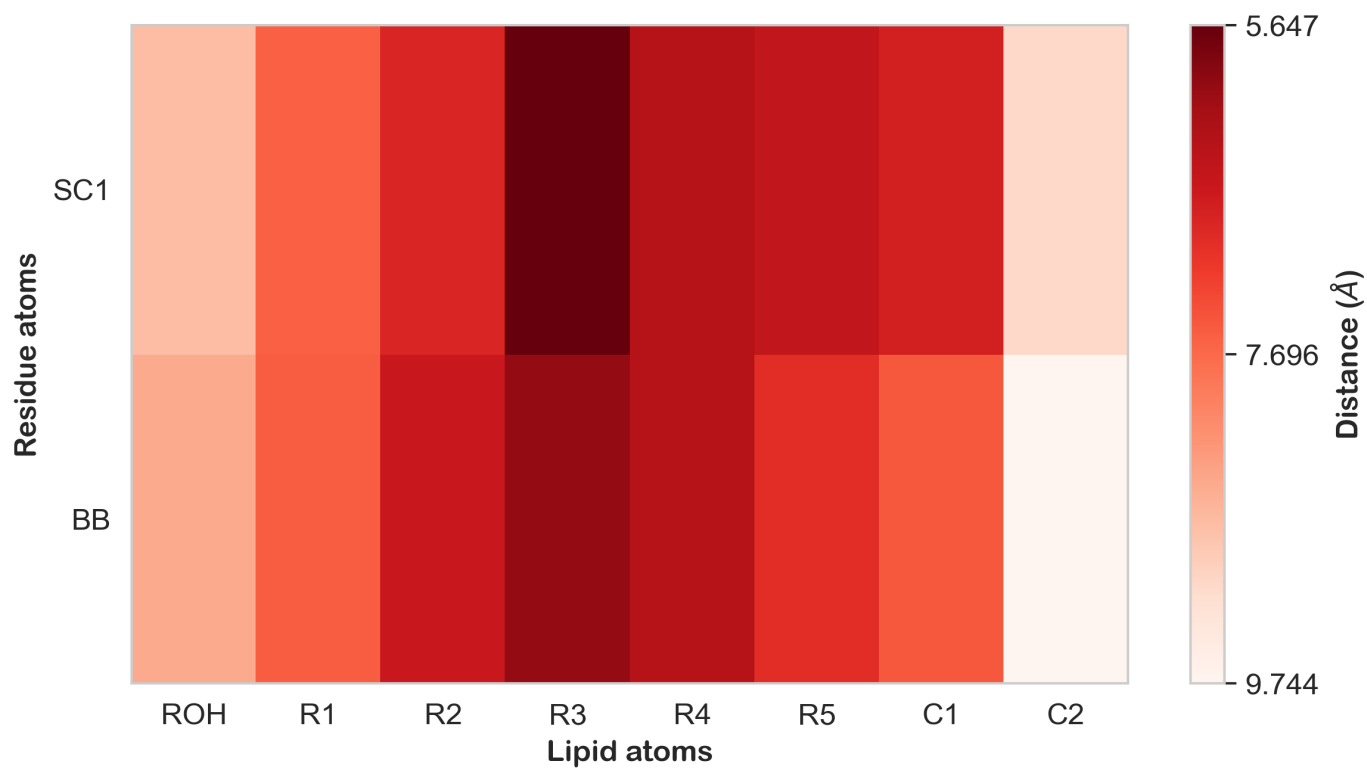
4. Analyzing interactions at the atomistic level

Interaction matrix

```
1 from prolint2.plotting import InteractionHeatMap
2
3 IHM = InteractionHeatMap(u, contacts)
4 IHM.create_plot(residue_id=125, lipid_id=3084, palette='Reds')
```

Analysis workflow with ProLint2

Interactions matrix between LEU 125 and CHOL 3084



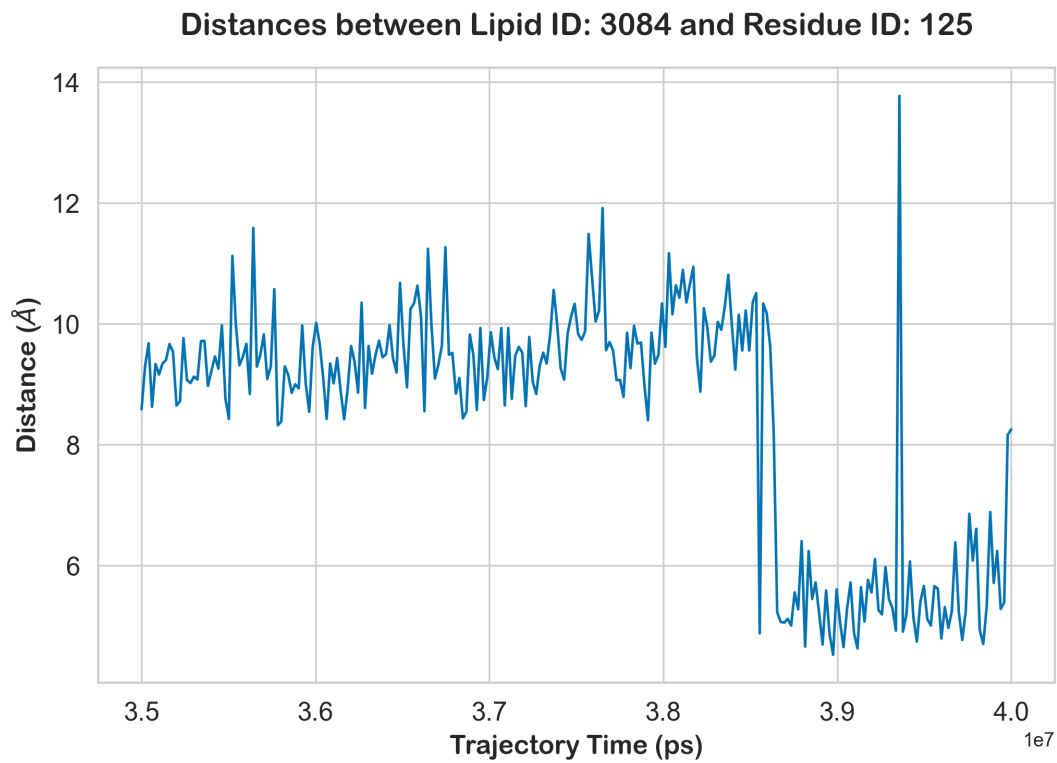
Analysis workflow with ProLint2

4. Analyzing interactions at the atomistic level

Distance evolution between specific atoms in residues

```
1 from prolint2.plotting import TwoPointDistanceEvolution
2
3 TPD = TwoPointDistanceEvolution(u)
4 TPD.create_plot(
5     lipid_id = 3084,
6     residue_id = 125,
7     lipid_atomname = 'ROH',
8     residue_atomname = 'BB',
9     unit = 'time',
10    smooth_line = True)
```


Analysis workflow with ProLint2



And More...

- Generate scripts automatically from the Plotters.

```
1 Plotter.generate_script('script_name.py')
```

- Calculation of Koff and residence times.
- Usage from the command-line interface.
- Interactive dashboard for ProLint server.



ProLint
Automated analysis and visualization of lipid-protein interactions.
<https://prolint.ca>  @bisejdiu



