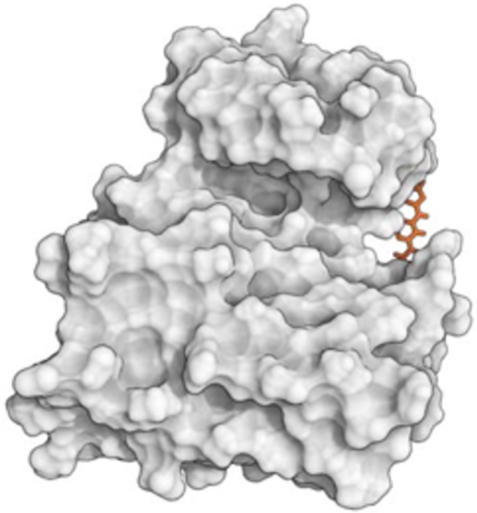
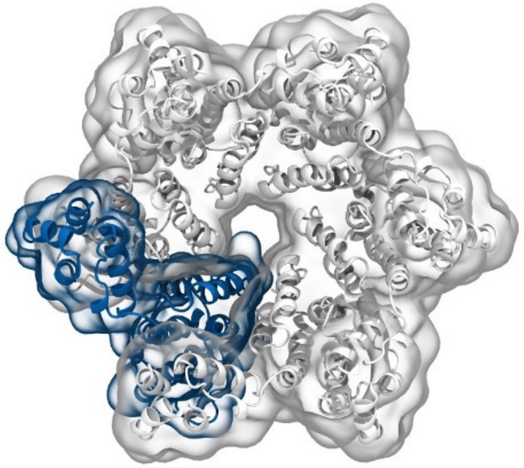
Simulation of Biomolecules



Lecture 5:
Introduction to MDAnalysis



2023 CCP5 Summer School

Dr Matteo Degiacomi
Durham University
matteo.t.degiacomi@durham.ac.uk

Dr Antonia Mey
University of Edinburgh
antonia.mey@ed.ac.uk

General information on MDAnalysis

Lecture material adapted from: Dr Micaela Matta, King's College London,

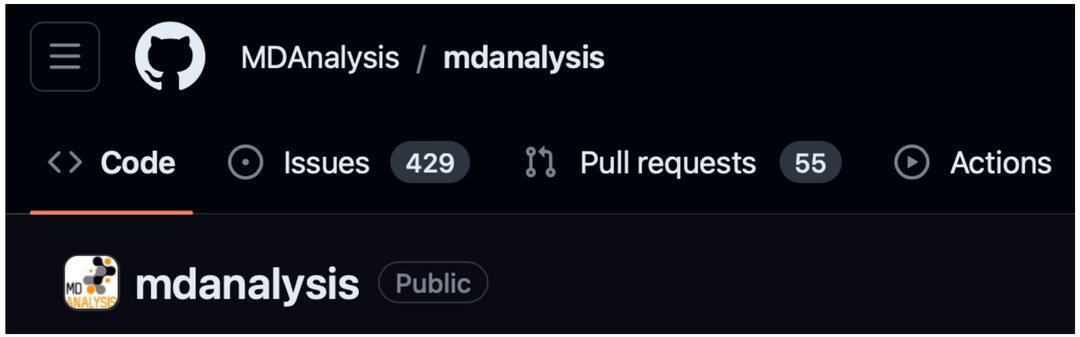
Email: micaela.matta@kcl.ac.uk

Join the active community with requests and issues

MD ANALYSIS Tutorial

MDAnalysis Tutorial

MDAnalysis version:	⩾ 0.18.0
Tutorial release:	3.0.0
Last updated:	Aug 19, 2020



https://www.mdanalysis.org/MDAnalysisTutorial/

https://github.com/MDAnalysis/mdanalysis

Session schedule

The session will be split into two sections:

- **Section 1:** Basics of MDAnalysis
  - Fundamental MDAnalysis objects
  - Atom selections
  - Visualising systems
  - Accessing bond, angle, and dihedral information

- **Section 2:** Positions, distances, and trajectory
  - Using position data
  - Calculating distances, bonds and angles
  - Accessing trajectory data

Getting started with MDAnalysis

Installing MDAnalysis

A conda environment containing all the dependencies you will need for this workshop is provided under `environment.yml`.

Should you want to install MDAnalysis under a separate environment, you can find the installation instructions here: https://www.mdanalysis.org/pages/installation_quick_start/. Installation is normally done through **pip** or **conda**.

```
pip install MDAnalysis
```

If you want to use the example data used here, you'll also need MDAnalysisTests:

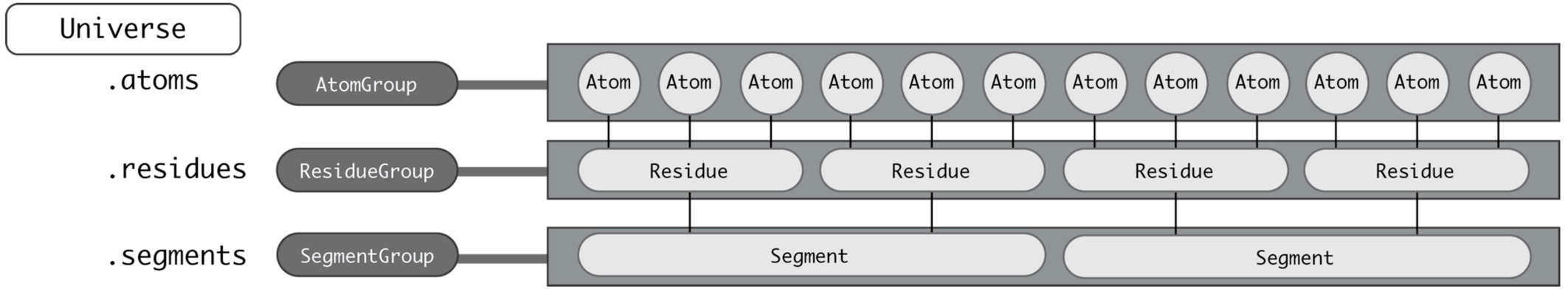
```
pip install MDAnalysisTests
```

MDAnalysis also has a repository of large example data files under MDAnalysisData:

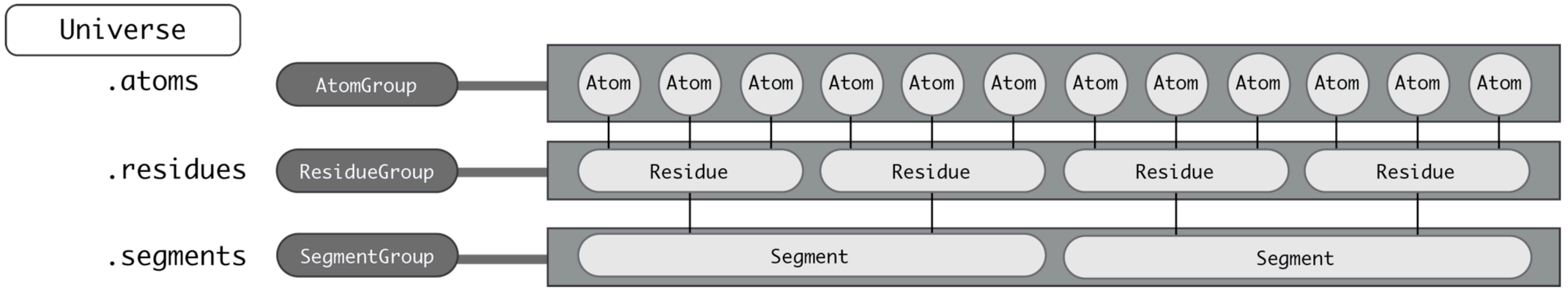
```
pip install MDAnalysisData
```

The general object structure of MDAnalysis

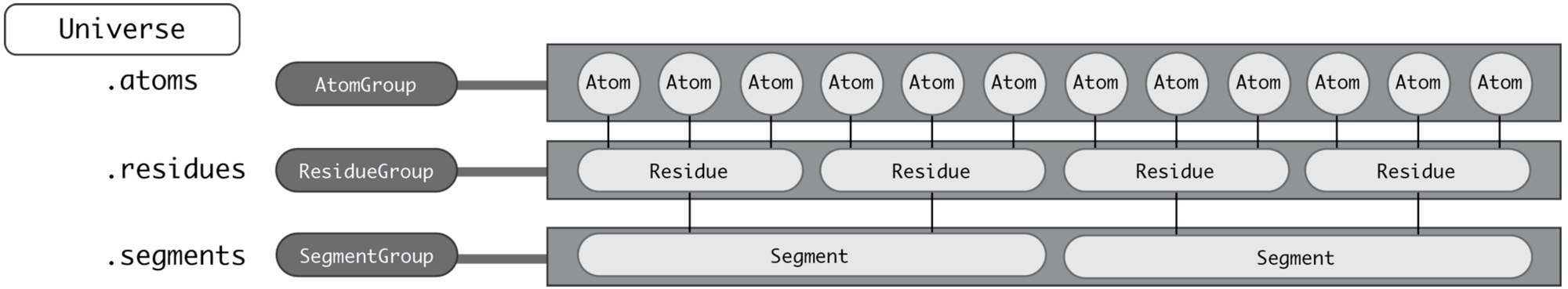
The two fundamental classes of MDAnalysis are the `Universe` and the `AtomGroup`.



- **The `Universe` contains everything about a molecular dynamics system**
  - Static information: atoms and their connectivities
  - Dynamic information: The trajectory



- The atoms in a `Universe` can be accessed through a hierarchy of containers:
  - *Atoms* can be grouped together into **an `AtomGroup`**
    - *Residues* are made up of *atoms*. They can be grouped into `ResidueGroups`
      - *Segments* are made up of *residues*. They can be grouped into `SegmentGroups`.



A (very) basic workflow for an analysis in MDAnalysis:

1. import MDAnalysis
2. create a `Universe`
3. define an `AtomGroup`
4. collect position data
5. analyse!

The `Universe`

The basic command for loading a universe is:

```
u = mda.Universe(topology, trajectory)
```

- The *topology* file must contain the atom information
- The (optional) *trajectory* file(s) contains the positions of atoms with time.

Note that some files can double as both a *topology* and a *trajectory* (e.g. PDB files).

MDanalysis supports over 40 input file types

In [ ]:
```
# First we import MDAnalysis
import MDAnalysis as mda

# Let's get some example data
from MDAnalysis.tests.datafiles import PSF, DCD

# and now load our universe!
u = mda.Universe(PSF, DCD)
u
```

Key properties of a `Universe`:

- `atoms`: an `AtomGroup` containing all of the system's atoms
    - similarly, `segments` and `residues`; a `SegmentGroup` and a `ResidueGroup`, respectively
- Various bond and angle information, as `TopologyGroups`: `bonds`, `angles`, `dihedrals`, `impropers` (if found in the topology file)
- `trajectory` (section 2): accessing time-dependent data structures

In [ ]:
```
u.bonds
```

AtomGroups

An `AtomGroup` is an "array" of atoms.

We can get various properties of each atom contained in an `AtomGroup` through attribues, e.g.:

- `names`
- `resnames`
- `resids`
- `charges`
- `masses`

Exactly which properties you can get depend on what is read from the topology (see the documentation)

In [ ]:
```
# Calling atoms creates an AtomGroup based
# on all the atoms in the system
ag = u.atoms
print(type(ag))
ag.names
```

Atom selections

We don't ususally want to work with the whole set of atoms in a trajectory. We need a way to create `AtomGroups` containing selected atoms.

Selection strings and `select_atoms`

We can use the `select_atoms()` method of an `AtomGroup` or `Universe` to return an `AtomGroup` based on a selection string.

There's a lot of options for selection strings (see the UserGuide)); including:

- selection by attribute (e.g. residue name (`resname`)), including presets like `protein`
- wildcard matching (`*`)
- boolean operators (`and`, `or`, `not`)
- geometric (e.g. `around`, `sphzone`, ...)
- and more!

In [ ]:
```
ag = u.select_atoms('protein')
view_ag = nv.show_mdanalysis(ag)
view_ag
```

In [ ]:
```
view_ag.add_licorice()
```

Working with coordinates

The most useful attribute of out atoms are their coordinates, available in the `positions` attribute of an `AtomGroup`

The positions are returned as a NumPy array, which we can then readily manipulate.

There are some built-in functions based on position data, e.g. `center_of_mass()`, `center_of_geometry()`

In [ ]:
```
pos = u.atoms.positions
print(pos)
```

This is just data from one frame - in the next section we will cover how to work with trajectories to get data across a whole simulation.

Built-in Analyses

- MDAnalysis has plenty of built-in analysis methods (RMSD, RMSF, MSD, PCA, PSA, etc...)
- These use AnalysisBase objects and can be called via a `run()` method to get data over the length of a trajectory.
- We will touch upon these during the next few notebooks, but they won't be a primary focus of this workshop.

In [ ]:
```
from matplotlib import pyplot as plt
from MDAnalysis.analysis.rms import RMSD
%matplotlib inline

u = mda.Universe(PSF, DCD)

c_alphas = u.select_atoms('name CA')

R = RMSD(c_alphas, c_alphas)
R.run()
plt.plot(R.results.rmsd.T[0], R.results.rmsd.T[2])
```