nglview

Туториал

Структурная биоинформатика, ФББ МГУ, 4 курс, осенний семестр

Ева Смородина

Что такое nglview?

Виджет IPython/Jupyter для интерактивного просмотра молекулярных структур и траекторий. Использует встраиваемый NGL Viewer для рендеринга. Поддерживает отображение данных из файлов, RCSB PDB, simpletraj и объектов библиотек анализа (MDTraj, PyTraj, MDAnalysis, ParmEd, RDKit, ASE, HTMD, biopython, cctbx, pyrosetta, schrodinger.structure).

Ссылки на репозиторий, документацию и полезное:

- https://github.com/nglviewer/nglview
- http://nglviewer.org/ngl/gallery/index.html
- http://nglviewer.org/nglview/release/v0.6.2.3/index.html#
- https://europepmc.org/article/PMC/6031024

Ссылка на Jupyter Notebook этого туториала:

- https://github.com/eva-smorodina/structural_bioinformatics

С чего начать работу с nglview?

Работает с Python 3.

- Установка с помощью conda:

  conda install nglview -c conda-forge (может понадобиться jupyter-nbextension enable nglview --py --sys-prefix)

- Установка с помощью pip:

  pip install nglview

  jupyter-nbextension enable nglview --py --sys-prefix

- Обновление установленного nglview:

  conda upgrade nglview --force

С чего начать работу с nglview?

Для работы с nglview в Jupyter Lab потребуется установить правильное расширение:

- **jupyter labextension install nglview-js-widgets**
  (https://github.com/arose/nglview/blob/master/devtools/nglview-jupyterlab.sh)

Подробнее об установке написано здесь:

- https://github.com/nglviewer/nglview

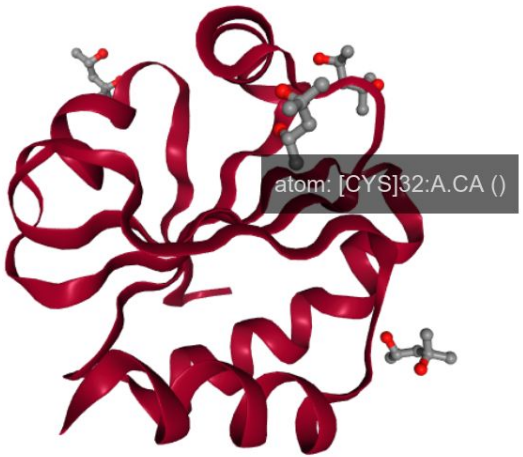
Первые шаги

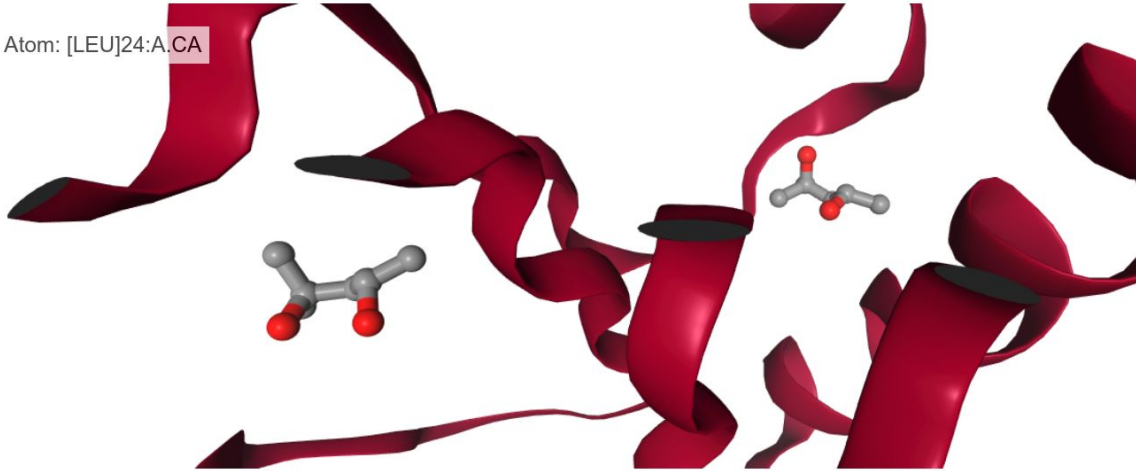
In [7]:
```
# Импорт nglview

import nglview as nv
```

In [8]:
```
# Загрузить структуру по PDBID из RCSB PDB и показать в виджете

view = nv.show_pdbid("2trx")
```

In [9]:
```
# Структуру в виджете можно поворачивать, приближать и перемещать с помощью мышки. При наведении мышки на
# структуру возникает интерактивное описание (название) объекта под мышкой. Если нажать на какой-то объект
# структуры, он перестанет быть интерактивным, и его описание появится в верхнем левом углу.

view
```

Первые шаги





Первые шаги

In [10]:
```
1 # Загрузить структуру из базы данных nglview и показать в виджете
2
3 view = nv.show_structure_file(nv.datafiles.PDB)
4 view
```

In [ ]:
```
1 nv.datafiles.
```
ALA3
ASE_Traj
GRO
MODULE_DIR
os
PDB
TRR
XTC

В nv.datafiles.PDB лежит тестовая PDB-структура для освоения возможностей NGLview. Также в nv.datafiles можно найти другие данные (аланиновые трипептид, GRO-файл, траектория, ...)

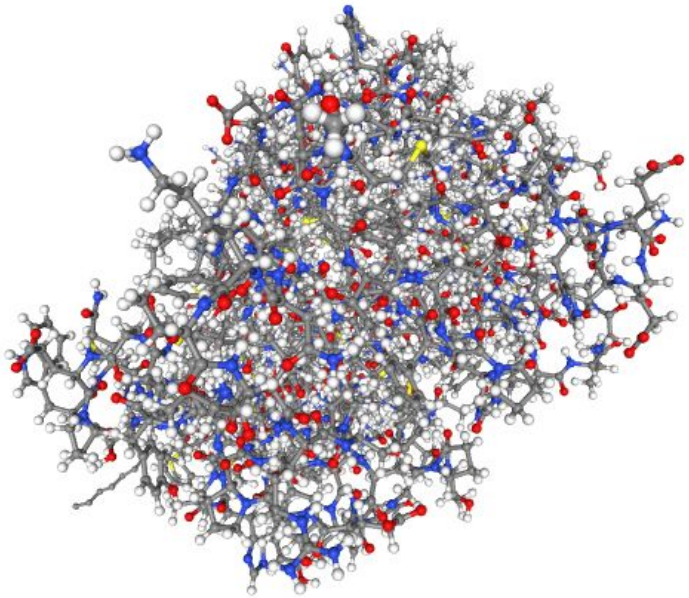
In [5]:
```
1 # Все изменения, которые будут вносится с отображение объекта, будут показаны в
2 # переменной, в которую изначально была загружена структура или траектория
```

Первые шаги

In [6]:
```
# Убрать cartoon из отображение

view.remove_cartoon()
```

Изменение способа отображения

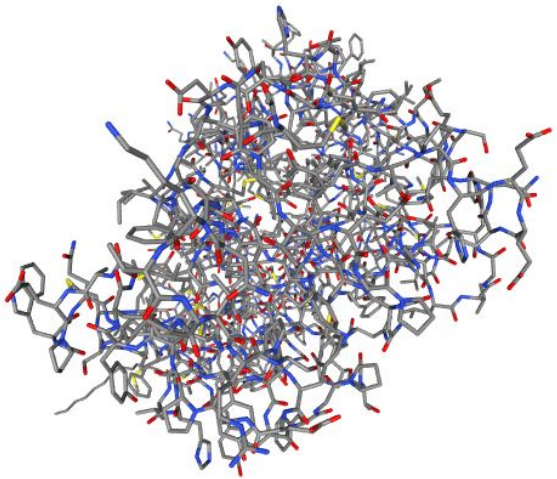
In [8]:
```
# Добавить шаро-стержневой вид структуры к отображению

view.add_ball_and_stick()
```



Изменение способа отображения

In [9]:

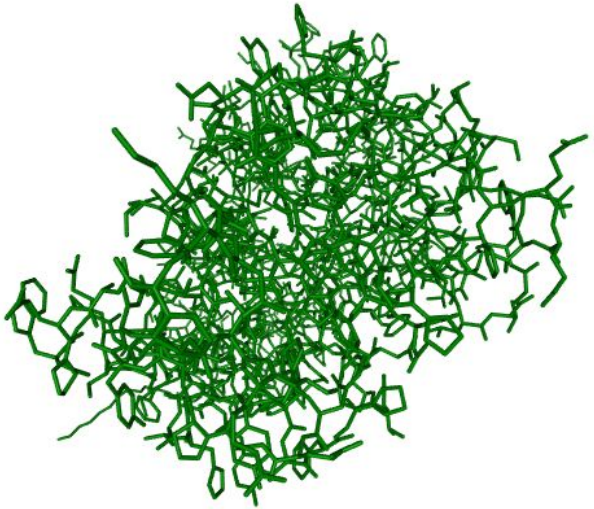
```
# Очистить отображение структуры (убрать имеющиеся параметры)

view.clear_representations()

# Отобразить структуру в виду шаро-стержневой моделе без водородов

view.add_licorice('not hydrogen')
```



Изменение способа отображения

In [10]:

```
# Очистить отображение структуры, отобразить структуру в виду
# шаро-стержневой моделе без водородов и покрасить все элементы
# в синий цвет

view.clear_representations()
view.add_licorice('not hydrogen', color='green')
```



Изменение способа отображения

In [11]:
```
# Помимо простого изменения цвета (предыдущий фрейм) можно
# изменять цвета по схеме (color sheme): atomindex, bfactor,
# chainid, chainindex, chainname, densityfit, electrostatic,
# element, entityindex, entitytype, geoquality, hydrophobicity,
# modelindex, moleculetype, occupancy, random, residueindex,
# resname, sstruc, uniform, value, volume (color='color sheme')
```

In [12]:
```
view.clear_representations()
view.add_cartoon(color='hydrophobicity')
```

Изменение способа отображения

In [11]:
```
# Помимо простого изменения цвета (предыдущий фрейм) можно
# изменять цвета по схеме (color sheme): atomindex, bfactor,
# chainid, chainindex, chainname, densityfit, electrostatic,
# element, entityindex, entitytype, geoquality, hydrophobicity,
# modelindex, moleculetype, occupancy, random, residueindex,
# resname, sstruc, uniform, value, volume (color='color sheme')
```

In [13]:
```
view.clear_representations()
view.add_cartoon(color='sstruc')
```

Изменение способа отображения

In [11]:
```
# Помимо простого изменения цвета (предыдущий фрейм) можно
# изменять цвета по схеме (color sheme): atomindex, bfactor,
# chainid, chainindex, chainname, densityfit, electrostatic,
# element, entityindex, entitytype, geoquality, hydrophobicity,
# modelindex, moleculetype, occupancy, random, residueindex,
# resname, sstruc, uniform, value, volume (color='color sheme')
```

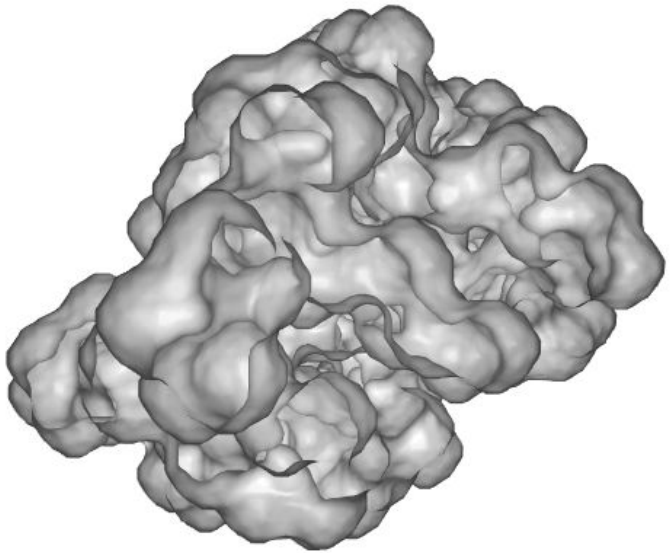
In [14]:
```
view.clear_representations()
view.add_cartoon(color='atomindex')
```

Изменение способа отображения

In [15]:
```
# Изменить параметры отображения cartoon: opacity (прозрачность),
# component (номер загруженного объекта, начиная с 0)

view.update_cartoon(opacity=0.4, component=0)
```

Изменение способа отображения

In [16]:
```
# Добавить отображение поверхности для CA атомов

view.clear_representations()
view.add_surface('.CA', opacity=0.3)
```

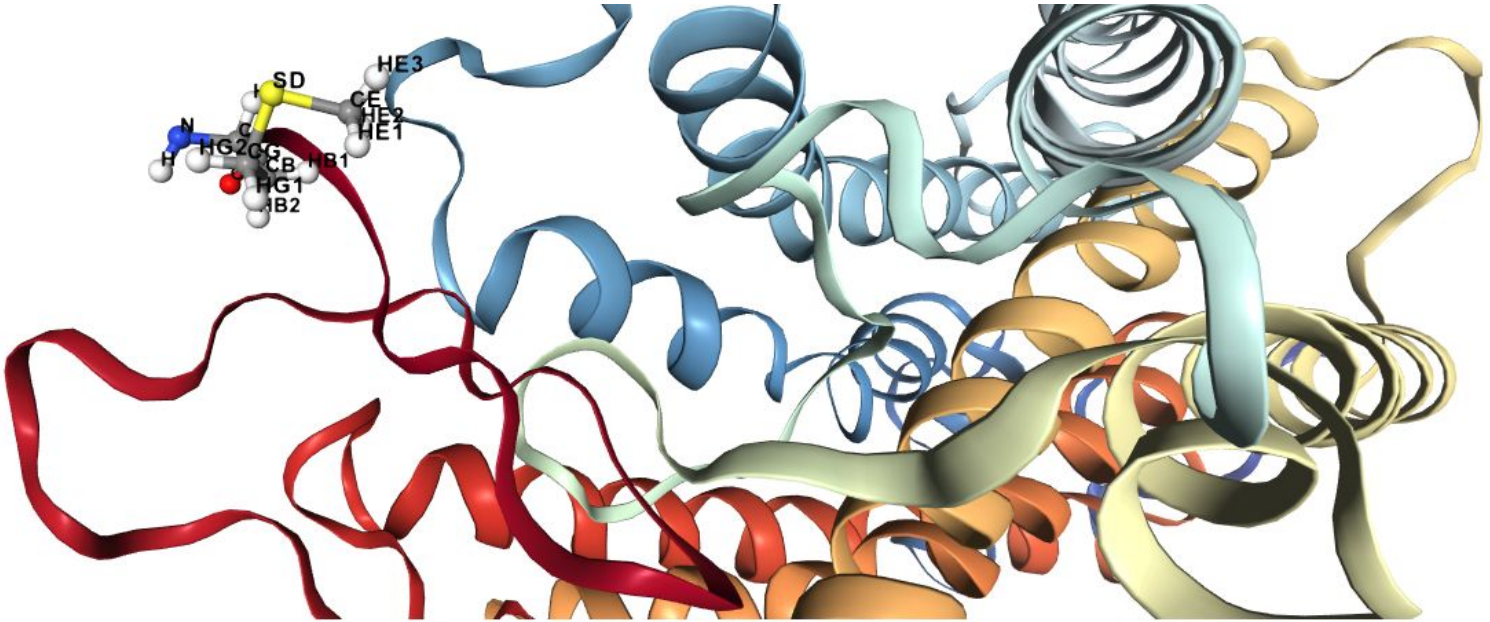


Изменение способа отображения

In [17]:

```
# Изменить отображение структуры, выделить 1-ый остаток объекта и
# подписать все атомы этого остатка

view.clear_representations()
view.add_cartoon(color='residueindex')
view.add_licorice('1')
view.add_ball_and_stick('1')
view.remove_label()
view.add_label('1', radius=1.5, color='black', label_type='atomname')
view

# Для лучшего рассмотрения изменений стоит покрутить объект и
# приблизить нужный остаток
```

Изменение способа отображения

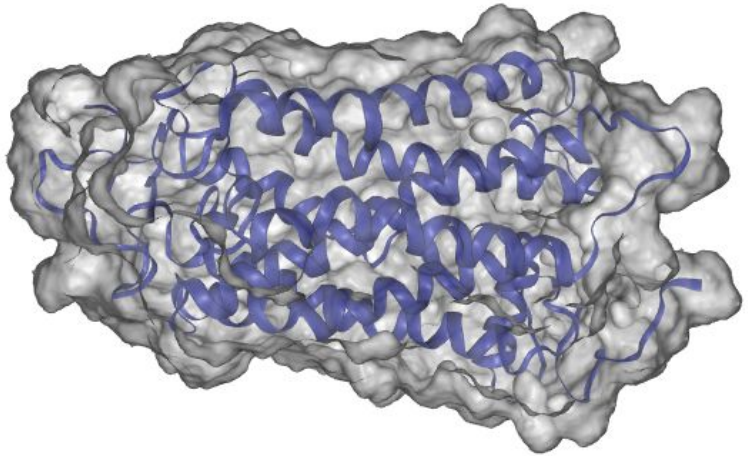


In [18]:
```
1 # Или можно использовать функции настойки ориентации объекта в пространстве,
2 # которые будут показаны ниже
```

Рендеринг для получения изображения

In [19]:

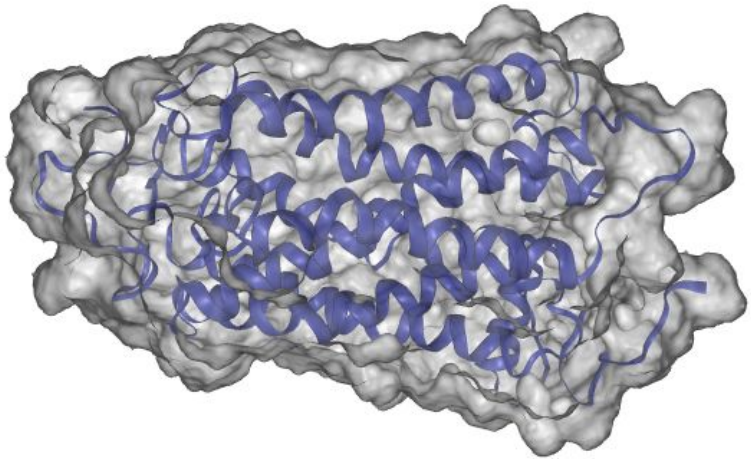
```
# Изменим отображения для получения более красивой картинки

view.clear_representations()
view.add_cartoon()
view.add_surface(opacity=0.3)
```



Рендеринг для получения изображения

In [23]:
```
# Показать меню ручного управления отображением

view._display_repr()
```

opacity 1.00 assembly default color_sche… wireframe isolevel 2.00

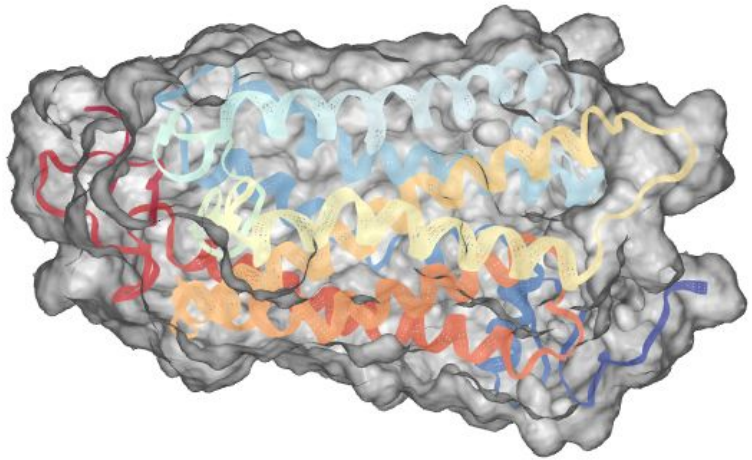


Рендеринг для получения изображения

In [27]:

```
# Показать меню ручного управления отображением

view._display_repr()
```

opacity 0.4 assembly default color_sche… atomindex wireframe isolevel 2.00



Ориентация объекта в пространстве

In [34]:
```
# Для воспроизведения точных фигур полезно сохранять ориентацию: это можно
# сделать, активировав синхронизацию камеры и получив/установив ориентацию

view._set_sync_camera()
```

In [35]:
```
# Получить ориентацию после вращения молекулы

view._camera_orientation

# Каждый раз при изменении положения объекта мышкой view._camera_orientation
# будет меняться
```

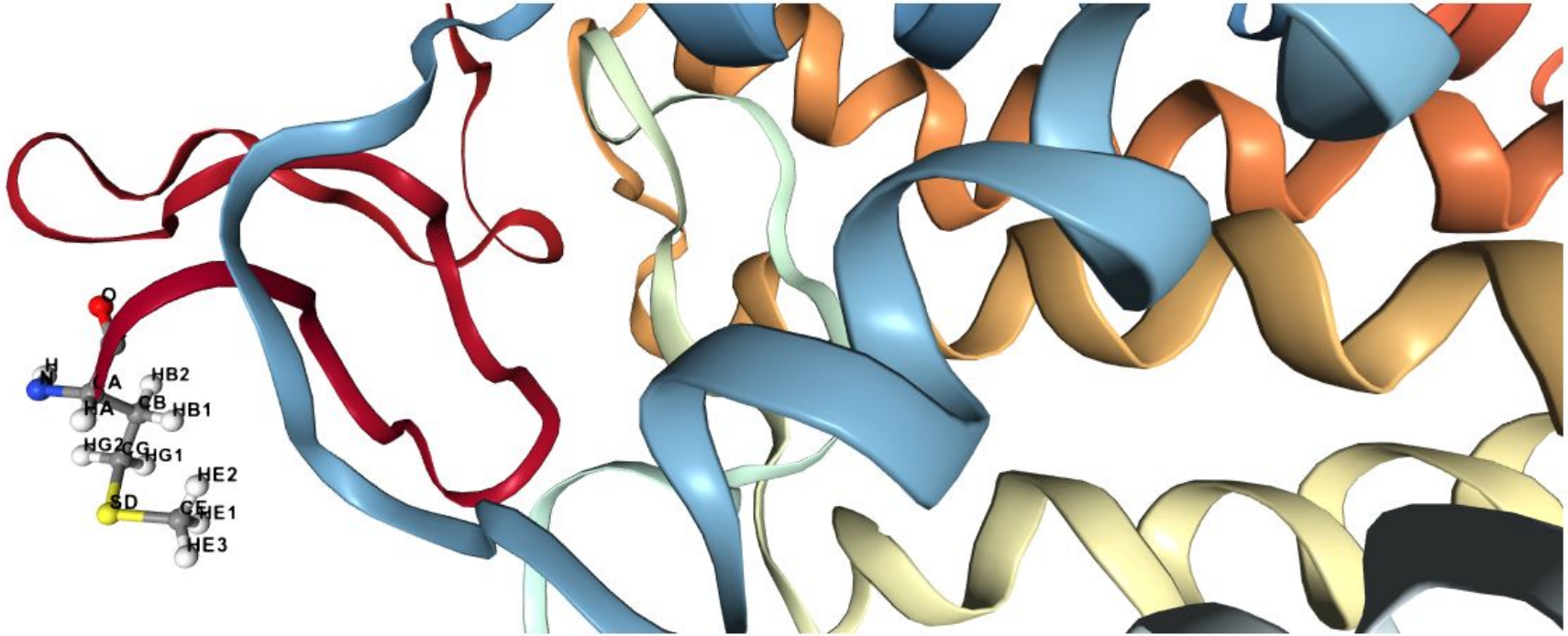
Out[35]:
```
[14.103716799412199,
 -7.80297819989794,
 99.69140055714205,
 0,
 -21.262861545709182,
 98.14170946455724,
 10.689818691308421,
 0,
 -97.70953143230508,
 -22.483216762219865,
 12.06354312352126,
 0,
 -37.284461975097656,
 -42.47041130065918,
 -44.55349826812744,
 1]
```

Ориентация объекта в пространстве

In [25]:

```
# Навести камеру на 1-ый остаток объекта (его ориентация была взята во время
# добавления названий атомов и подбора мышкой оптимального расположение объекта
# для рассмотрения этого остатка)

zoom_1 = [5.32662441137791,
 31.75442593787687,
 -7.85010941127703,
 0,
 1.9290071238099673,
 -8.24461265288586,
 -32.04131716633245,
 0,
 -32.6534334477954,
 4.692921008583665,
 -3.17340330099961,
 0,
 -34.74262940992502,
 -33.62137402595412,
 -22.393204617630467,
 1]

view._camera_orientation = zoom_1
view.clear_representations()
view.add_cartoon(color='residueindex')
view.add_licorice('1')
view.add_ball_and_stick('1')
view.remove_label()
view.add_label('1', radius=1.5, color='black', label_type='atomname')
view
```

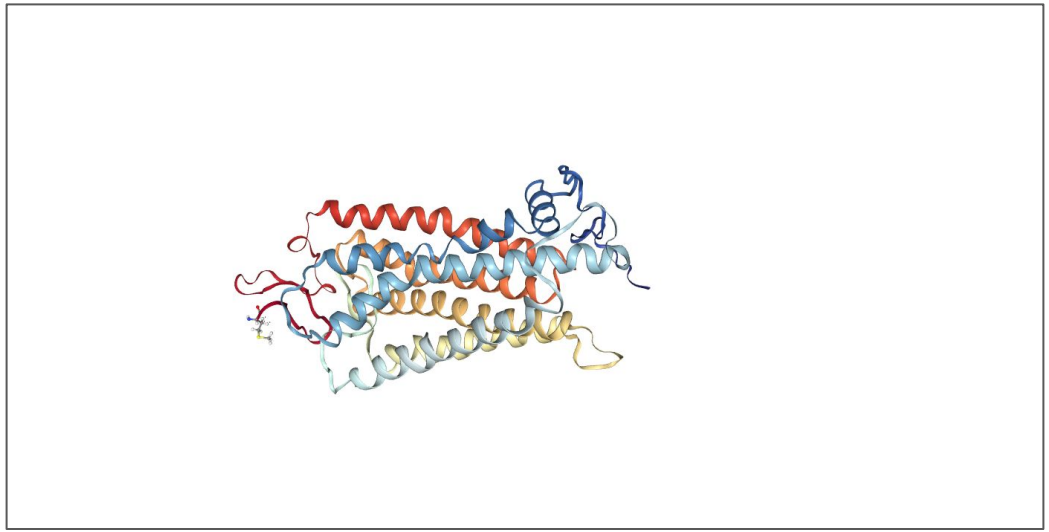
Ориентация объекта в пространстве



Ориентация объекта в пространстве

In [26]:

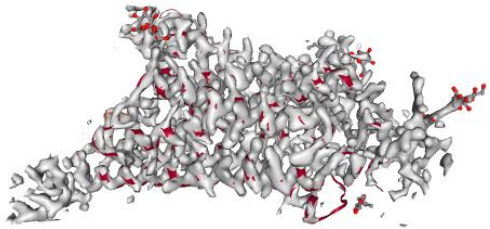
```
# Это можно использовать для получения фигур точного размера каждый раз

view._remote_call("setSize", target="Widget", args=["800px", "500px"])

# Центрировать и увеличить молекулу

view.center()
```



Работа с электронной плотностью

In [65]:
```
1 view = nv.show_structure_file('3pqr.pdb')
2 view.add_component('3pqr_2fofc.dsn6')
3 view
```

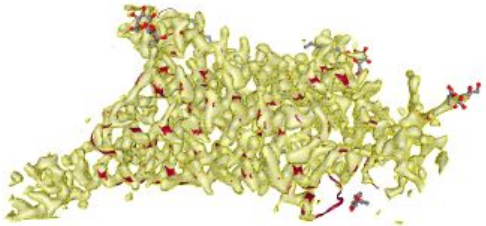
Можно загрузить PDB скаченного файла с помощью nv.show_structure_file. После этого нужно добавить новый компонент, в котором будет электронная плотность (в данном случае типа 2fofc)



Работа с электронной плотностью

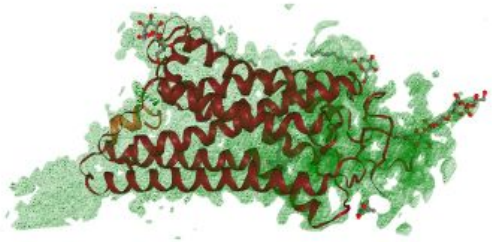
```
In [47]:  1  view.component_1.add_surface(opacity=0.4, wireframe=True, color='yellow')
```

Для отображения электронной плотности
нужно вызвать компонент 1 (т. к.
электронную плотность мы загрузили в него)
и добавить отображение в виде surface



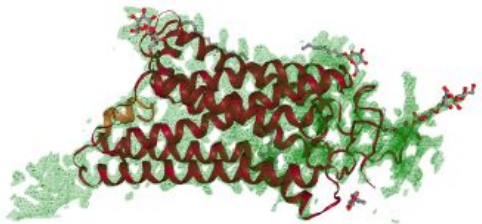
Работа с электронной плотностью

In [58]:
```
1 view.component_1.update_surface(isolevel=1.0, opacity=0.2, isolevelType='value', color='green')
2 view
```



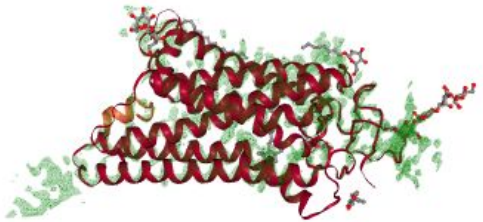
Работа с электронной плотностью

In [59]:
```
1 view.component_1.update_surface(isolevel=2.0, opacity=0.2, isolevelType='value', color='green')
2 view
```



Работа с электронной плотностью

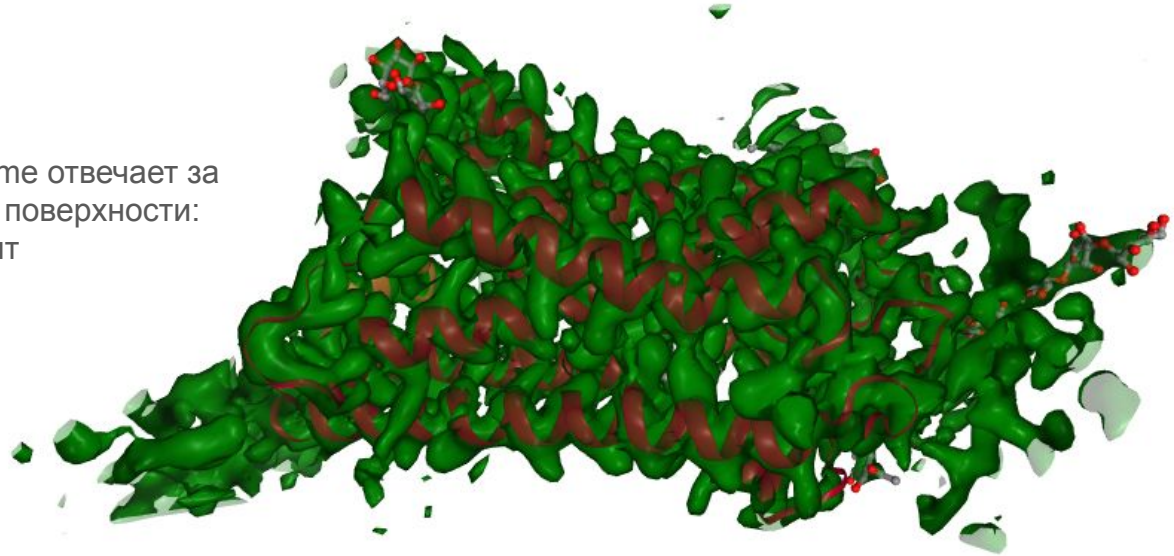
In [60]:
```
1 view.component_1.update_surface(isolevel=3.0, opacity=0.2, isolevelType='value', color='green')
2 view
```



Работа с электронной плотностью

In [63]:
```
1 view.component_1.update_surface(isolevel=1.0,opacity=0.2, isolevelType='value', color='green', wireframe=False)
2 view
```

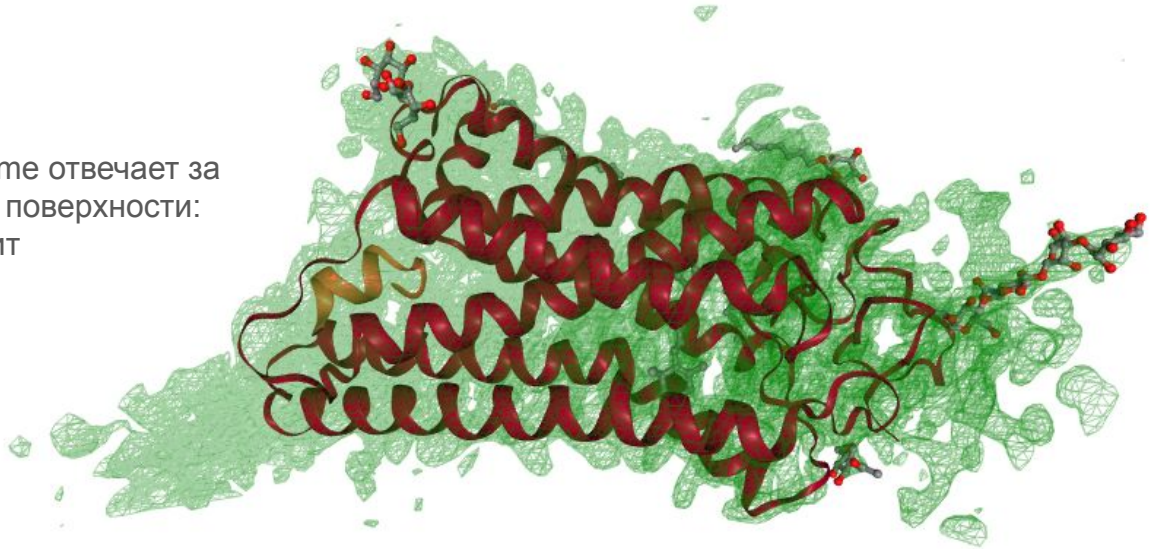
Параметр wireframe отвечает за тип отображения поверхности: сетка или монолит



Работа с электронной плотностью

In [64]:
```
1 view.component_1.update_surface(isolevel=1.0,opacity=0.2, isolevelType='value', color='green', wireframe=True)
2 view
```

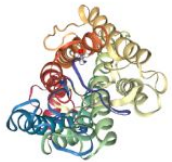
Параметр wireframe отвечает за тип отображения поверхности: сетка или монолит



Визуализация траекторий

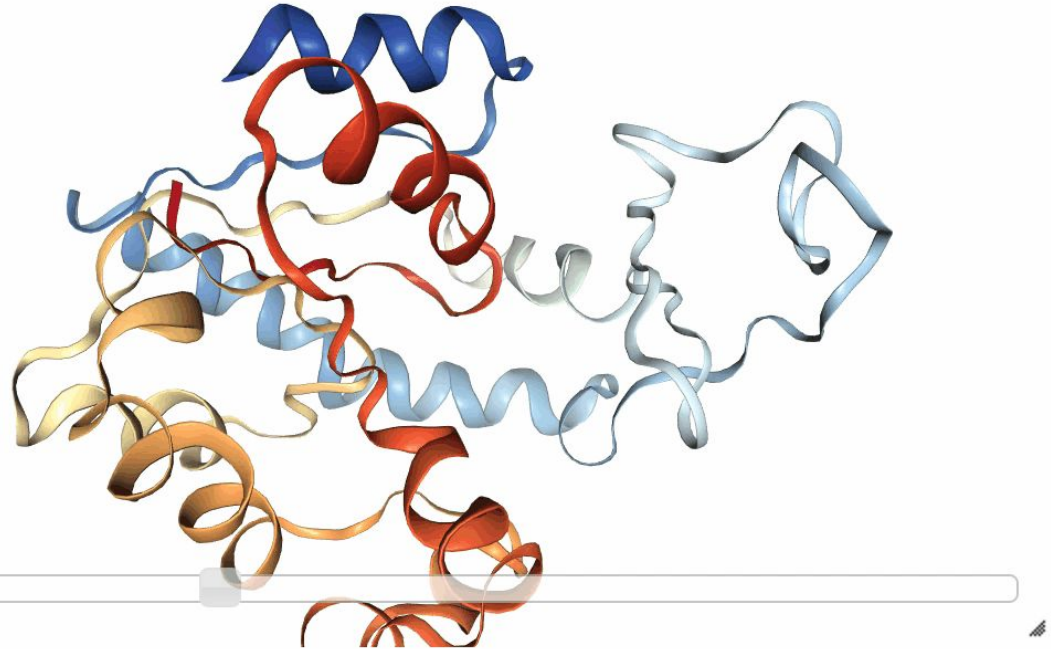
In [38]:

```
# Загрузить траекторию из базы данных nglview с помощью MDTraj

import warnings
warnings.filterwarnings('ignore', category=DeprecationWarning)
warnings.filterwarnings('ignore', category=UserWarning)

import mdtraj as md
m_traj = md.load(nv.datafiles.TRR, top=nv.datafiles.PDB)
m_view = nv.show_mdtraj(m_traj)
m_view

## Аналогично для PyTraj и MDAnalysis
#
# import pytraj as pt
# p_traj = pt.load(nv.datafiles.TRR, top=nv.datafiles.PDB)
# p_view = nv.show_pytraj(p_traj)
# p_view
#
# import warnings
# warnings.filterwarnings('ignore', category=DeprecationWarning)
# from MDAnalysis import Universe
# mda_traj = Universe(nv.datafiles.PDB, nv.datafiles.TRR)
# mda_view = nv.show_mdanalysis(mda_traj)
# mda_view
```

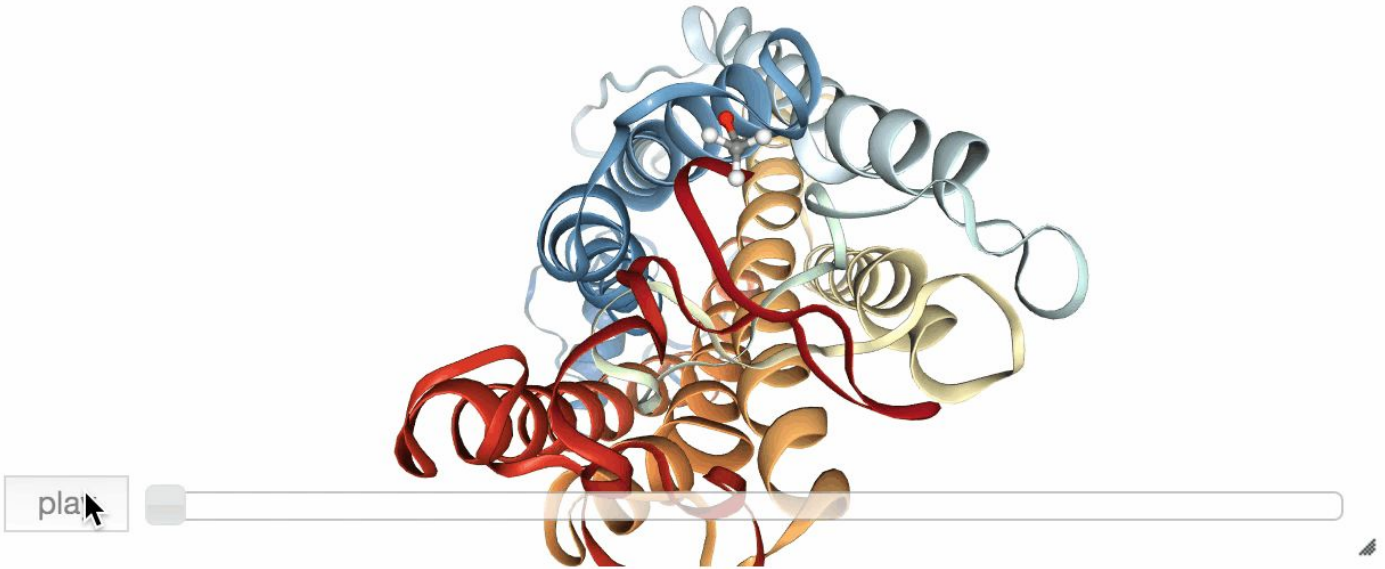


play

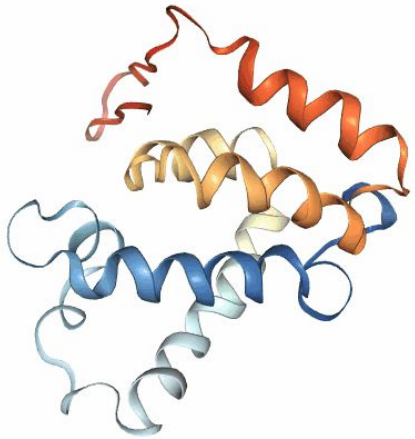
In [1]:
```
from MDAnalysis import Universe
from MDAnalysisTests import datafiles
import nglview as nv

u = Universe(datafiles.PSF, datafiles.DCD)

view = nv.show_mdanalysis(u)
view
```



In [1]:
```
import mdtraj as md
import nglview as nv

traj = md.load(nv.datafiles.TRR, top=nv.datafiles.PDB)

view = nv.show_mdtraj(traj)
view
```



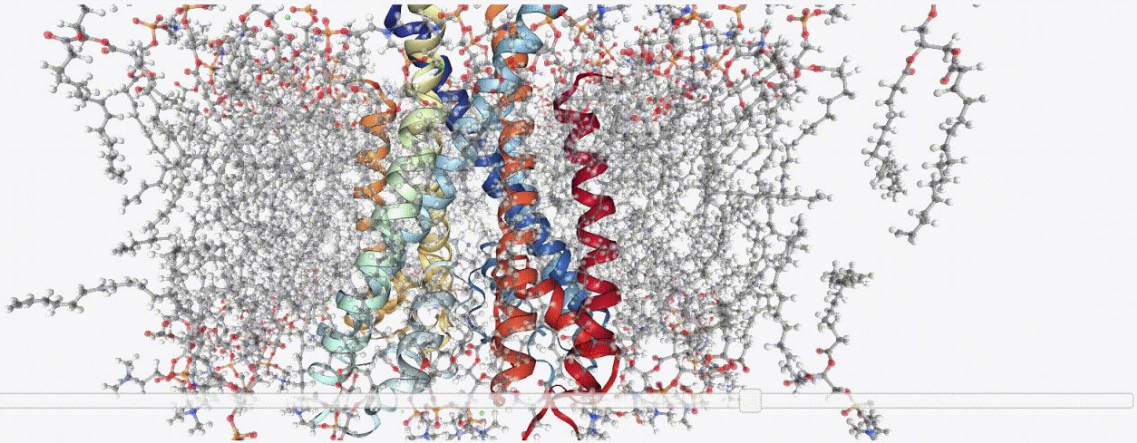
In [1]:
```
import parmed as pmd
import nglview as nv

# !wget http://files.rcsb.org/download/1G03.pdb.gz
parm = pmd.load_file('1G03.pdb.gz')
view = nv.show_parmed(parm)
view.player.delay = 150
view
```



play

In [1]:
```
import pytraj as pt
import nglview as nv
```

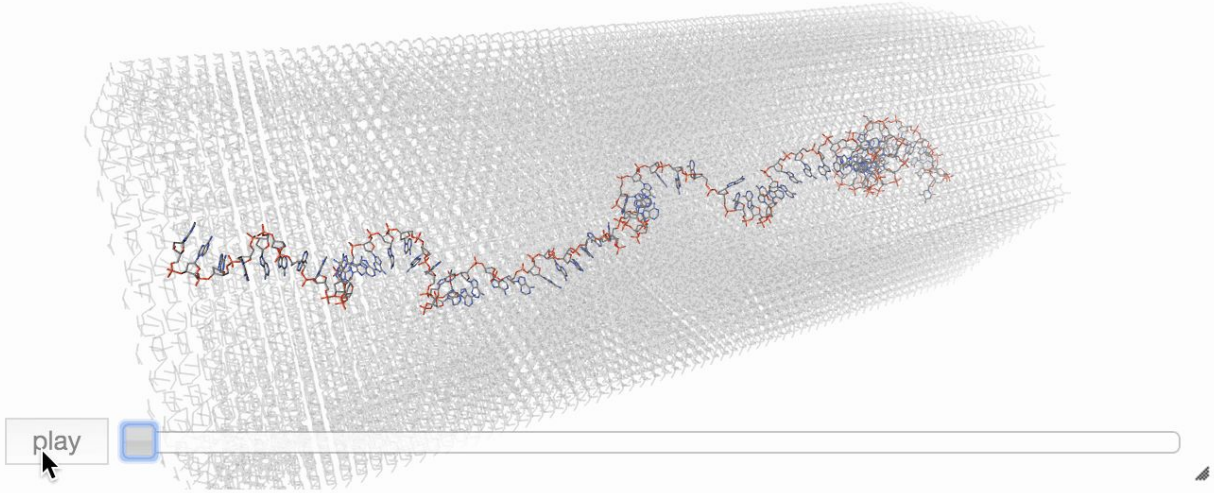
In [2]:
```
traj =  pt.load('sim.nc', top='sim.prmtop')
view = nv.show_pytraj(traj)
view
```



In [3]:
```
view.clear()
view.add_cartoon('protein', color_scheme='residueindex')
view.add_ball_and_stick('not protein', opacity=0.5)
```

```
pytraj.TrajectoryIterator, 31 frames:
Size: 0.135101 (GB)
<Topology: 194978 atoms, 47970 residues, 0 mols, non-PBC>
```

In [13]:
```
view = traj.view()
view
```



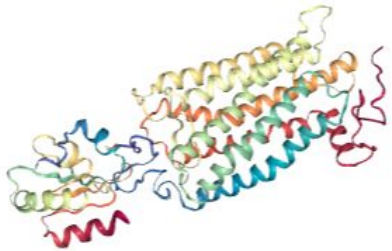
In [24]:
```
view.clear()
view.add_licorice('nucleic and not hydrogen')
view.add_line('not nucleic and not TIP', color='grey', opacity=0.3)
```

In [27]:
```
view.player.delay = 0.2
```

Добавление объектов и траекторий

In [28]:

```
from MDAnalysis import Universe
mda_traj = Universe(nv.datafiles.PDB, nv.datafiles.TRR)
view = nv.show_mdanalysis(mda_traj)
view

# Добавить траекторию или структуру к открытому в виджете объекту

m_traj = md.load('2trx.pdb')

view.add_trajectory(m_traj)
view
```



play

Добавление объектов и траекторий

In [46]:
```
# Показать количество компонент в виджете

view.n_components
```

Out[46]: 2

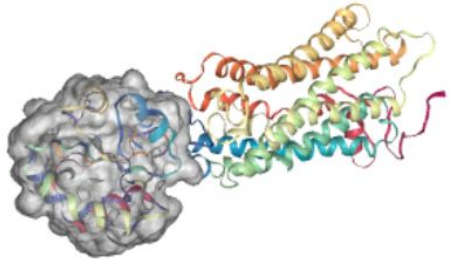
In [47]:
```
# Отобразить 1-ый компонент в виде cartoon. Доступ к компонентам
# существляется по индексам (view[1]) или view.component_n, где
# n - 0, 1, ...

view.component_1.add_cartoon()
```

In [48]:
```
# Добавить отображение поверхности к 1-ому компоненту

view.component_1.add_surface(opacity=0.3)
```

In [49]:
```
view.orientation = [-43.33923135864405,
 -115.77635240335854,
 61.38834848744958,
 0,
 16.54229893103012,
 59.30776982619632,
 123.53106032460101,
 0,
 -129.99632860938627,
 46.14549537080969,
 -4.746565670481519,
 0,
 -40.204999923706055,
 -42.970001220703125,
 -44.16499936580658,
 1]
```

Добавление объектов и траекторий

```
In [50]:  1  view
```



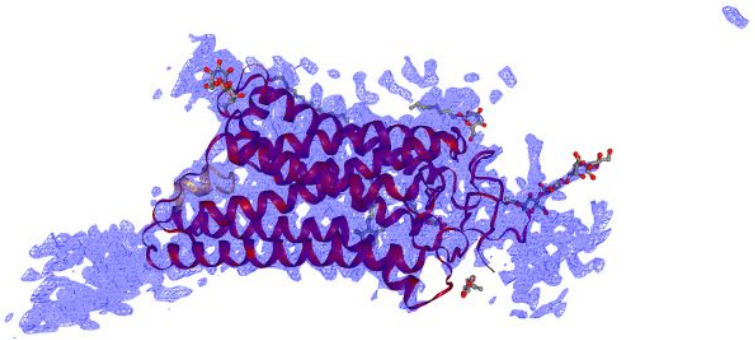
play

Примеры визуализации с nglview

In [51]:
```
1 traj = md.load('3pqr.pdb')
2 view = nv.show_mdtraj(traj)
3 view
```

Примеры визуализации с nglview

In [35]:
```
1 view.add_component('3pqr.ccp4.gz')
2 view.component_1.clear()
3 view.component_1.add_surface(opacity=0.4, wireframe=True, color='blue')
```



Примеры визуализации с nglview

In [8]:
```
# load pdb file
traj3 = pt.load('data/3pqr.pdb')

# create view
view3 = nv.show_pytraj(traj3)

# display
view3.center_view()
view3
```

In [9]:
```
# load ccp4 data
# use: view3._load_data

view3._load_data('data/3pqr.ccp4.gz')
```

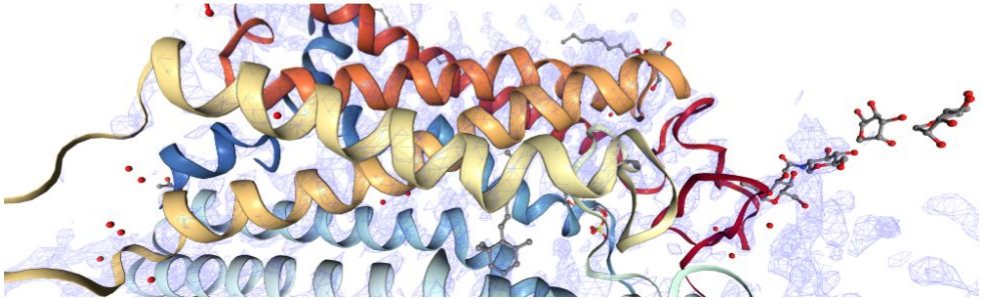
In [10]:
```
# add representations for loaded data (model 1)
view3._clear_repr(component=1)
view3.add_surface(component=1, color='blue', wireframe=True, opacity=0.2, isolevel=3.)
```

After zooming in by mouse, you should expect to see

In [11]:
```
view3.render_image()
```

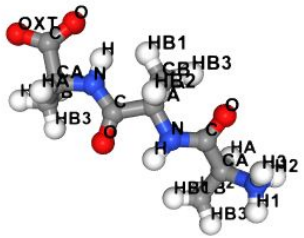
In [12]:
```
# need to call _display_image in different Cell
view3._display_image()
```

Out[12]:



Примеры визуализации с nglview

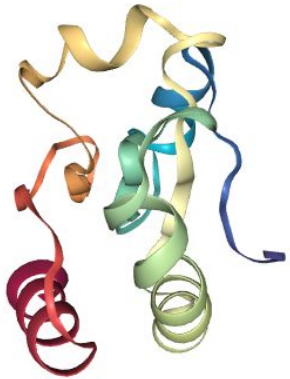
In [53]:
```
1 view = nv.show_structure_file(nv.datafiles.ALA3)
2 view.add_label(radius=2, color='black', label_type='atomname')
3 view
```



Примеры визуализации с nglview

In [54]:

```
1 traj = md.load('2trx.pdb')
2 view = nv.show_mdtraj(traj)
3 view
```

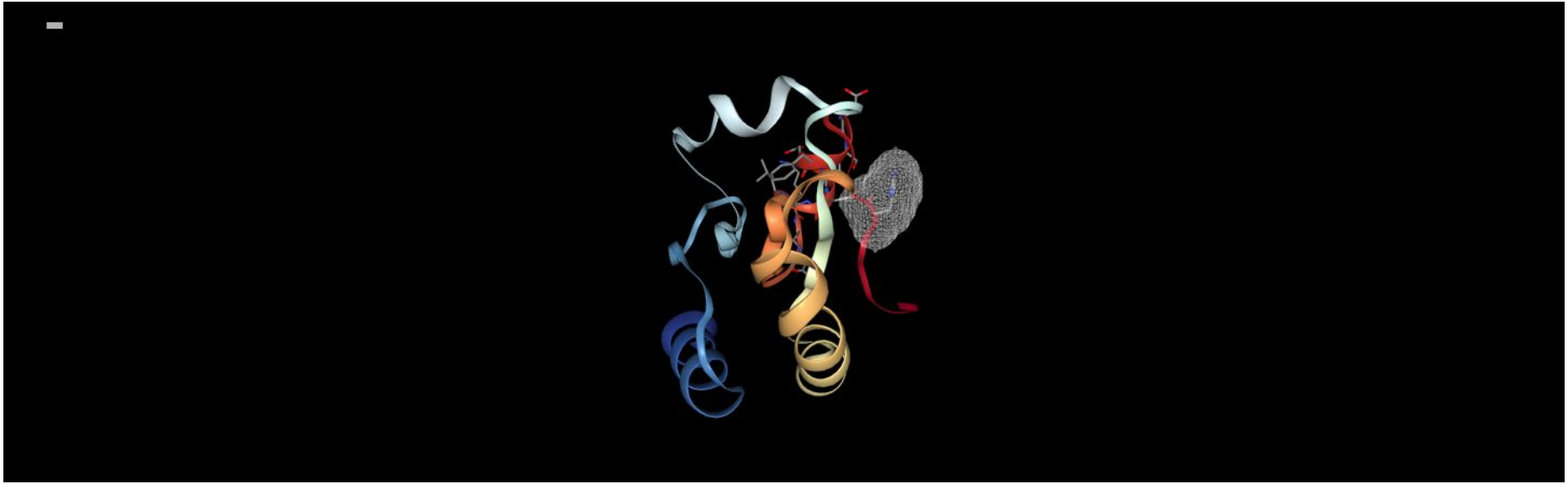


Примеры визуализации с nglview

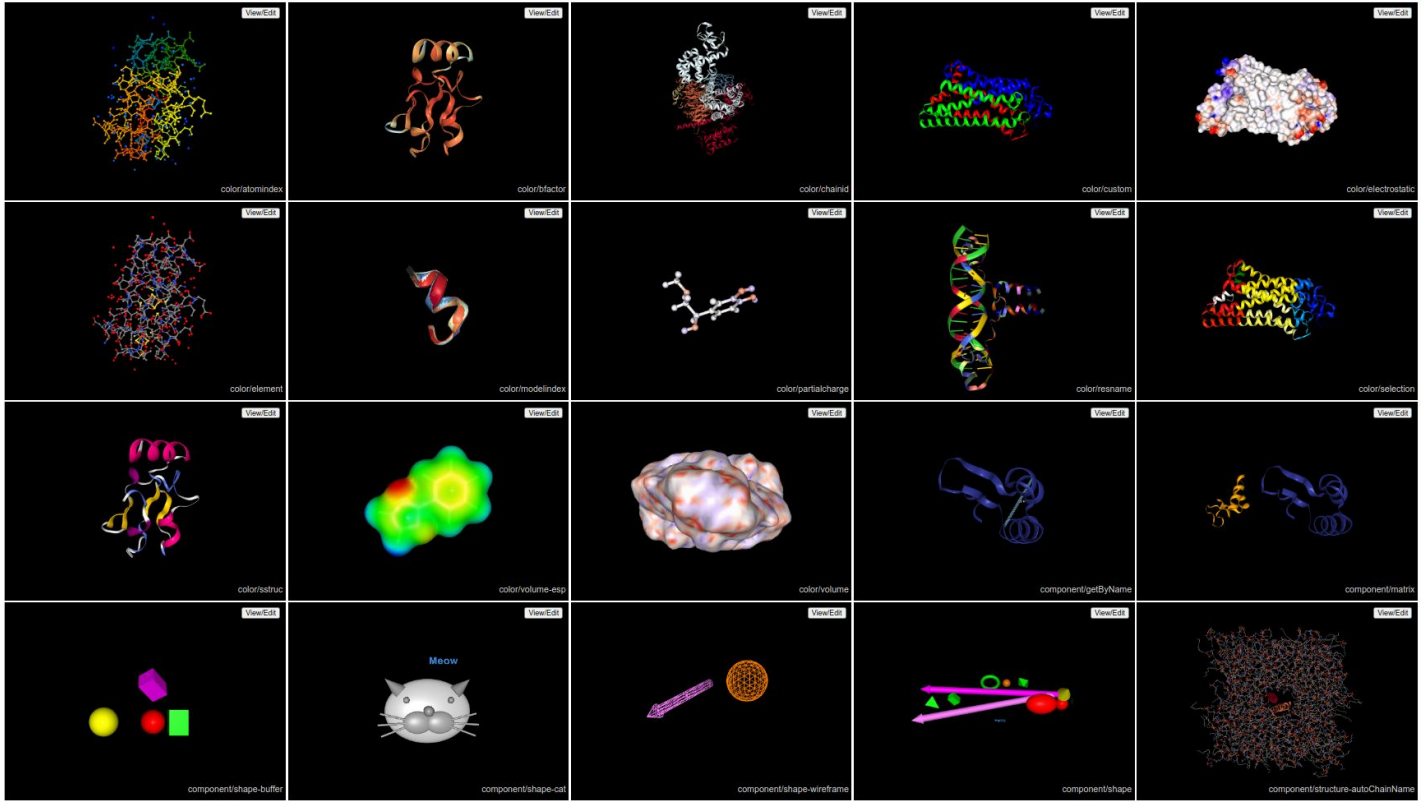
In [38]:

```
representations = [
    {"type": "cartoon", "params": {
        "sele": "protein", "color": "residueindex"
    }},
    {"type": "surface", "params": {
        "sele": "6",
        "opacity": "0.3",  "wireframe": True, "color": "white"
    }},
    {"type": "licorice", "params": {
        "sele": "6",
    }},
    {"type": "licorice", "params": {
        "sele": "10-20 and not hydrogen"
    }}
]

view.representations = representations
view.background = 'black'
```

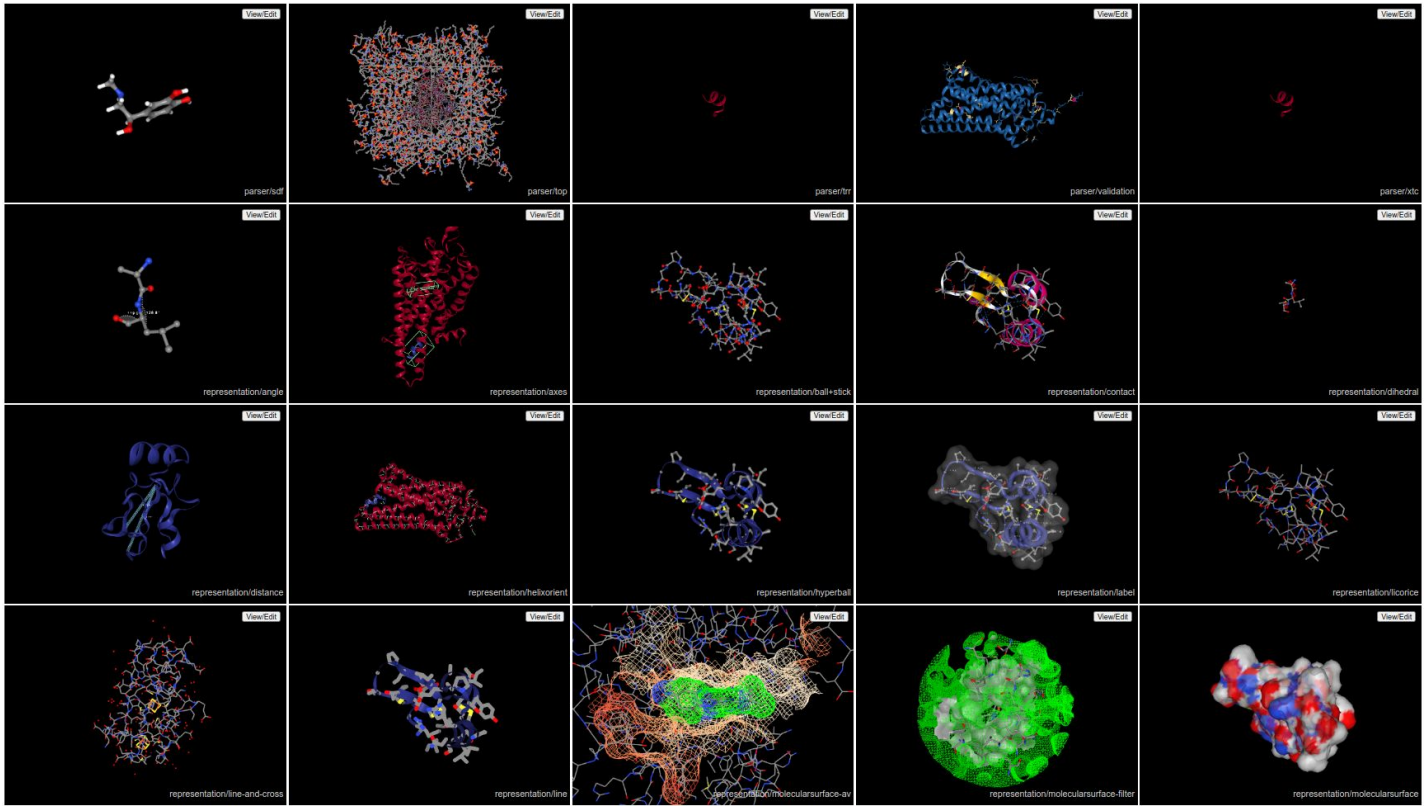
Примеры визуализации с nglview



Примеры визуализации с nglview



Примеры визуализации с nglview



Примеры визуализации с nglview