Overview_Figures_EncoderMaps

August 21, 2025

[1]:
```
import mdtraj as md
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
```

0.0.1 Load xy coordinates from the dimensionality reduction

Dimensionality reduction was performed in 3_EncoderMap_full_length_FAT10.ipynb

[2]:
```
low_d_projection = np.load("../EncoderMap_low_d_coordinates_full_length_FAT10.
↪npy")
```

0.0.2 Load closeness centralities

Centralities were calculated in 1_Calculating_Closeness_Centralities_full_length_FAT10.ipynb

[3]:
```
closeness = np.load("../Closeness_Centralities_full_length_FAT10.npy")
```

0.0.3 Load cluster IDs

HDBSCAN cluster IDs were calculated in 4_Clustering_EncoderMap_full_length_FAT10.ipynb

[4]:
```
cluster_ids = np.load("../Arcdiagrams/
↪1_Cluster_IDs_Encodermap_HDBSCAN_full_length_FAT10.npy")
```

0.0.4 Load PDB file for residue names

[5]:
```
PDB_file = md.load("../start_frame_FAT10.pdb")
residue_dict = {index:str(residue) for index, residue in enumerate(PDB_file.
↪topology.residues)}
residues = list(residue_dict.values())
```

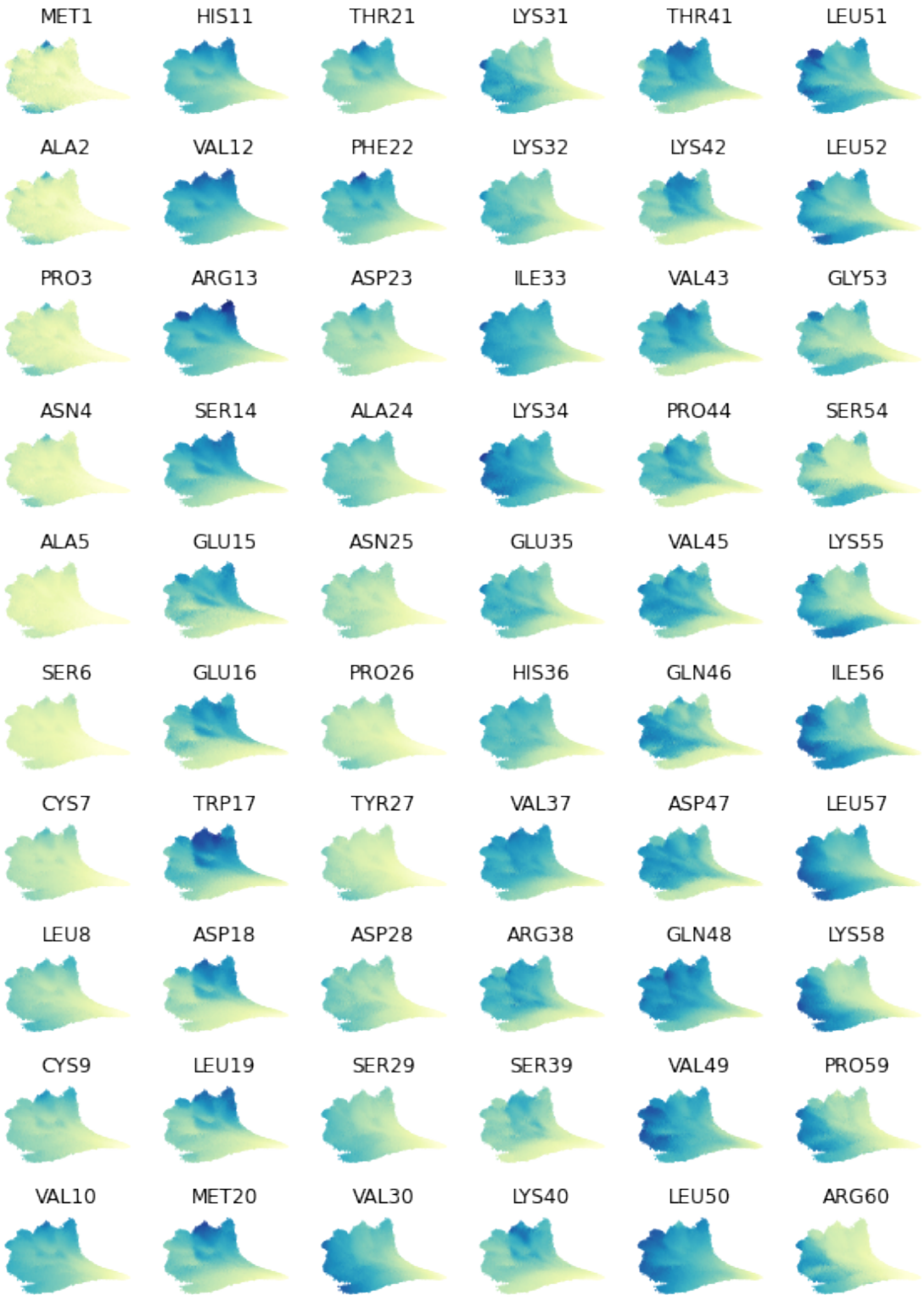
0.0.5 Indexing function for plotting

[6]:
```
def row_based_idx(num_rows, num_cols, idx):
    return np.arange(1, num_rows*num_cols + 1).reshape((num_rows, num_cols)).
 ↪transpose().flatten()[idx-1]
```

0.0.6 Plot EncoderMap colored by closeness centralities for all residues of FAT10

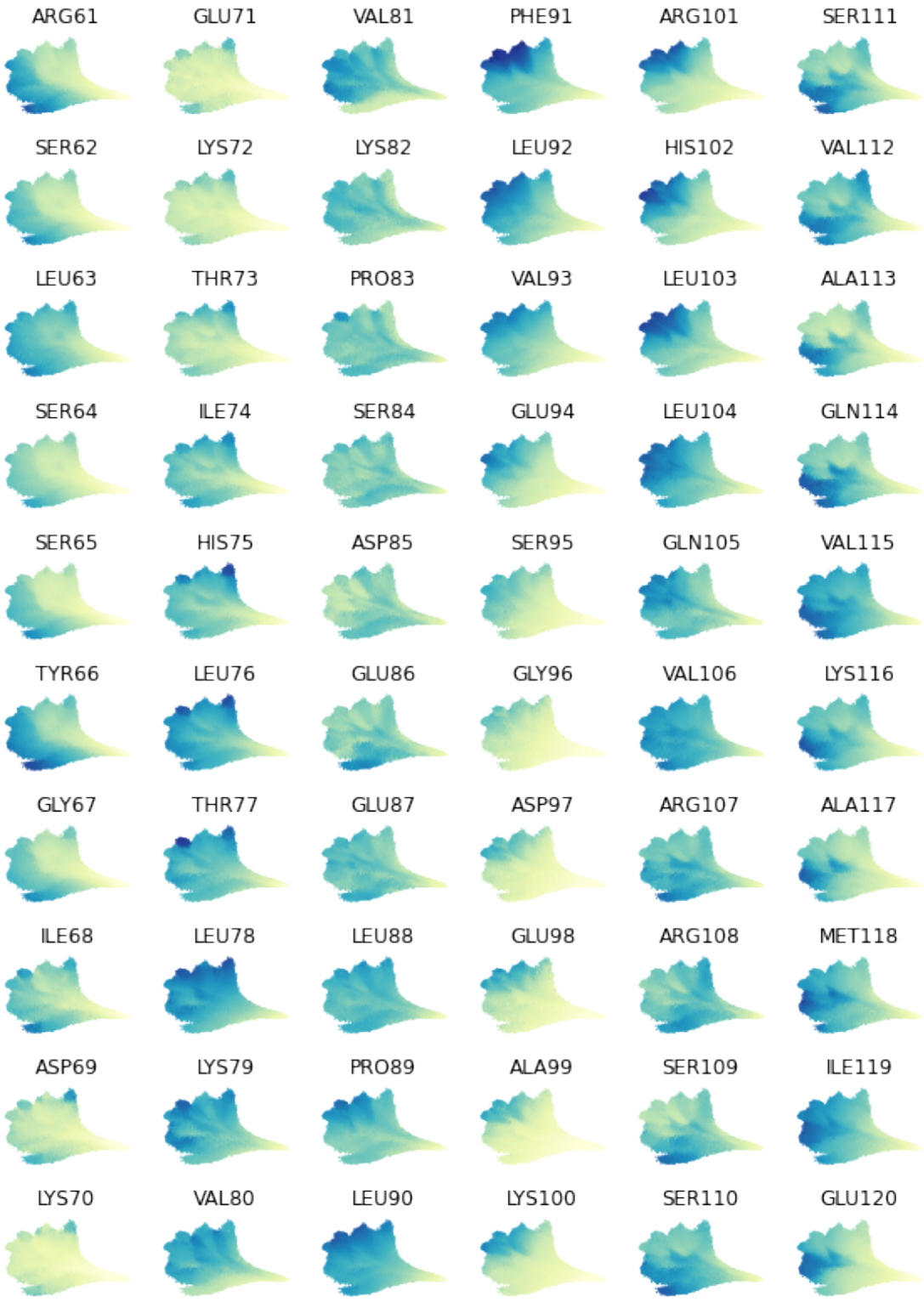
[7]:
```
%matplotlib inline
plt.rcParams.update({'font.size': 10})
fig = plt.figure(figsize = (8,11), tight_layout = True)
rows = 10
columns = 6
counter = 1

for panel_index, residue_index in enumerate(range(0,60)):
    ax =plt.subplot(rows, columns, row_based_idx(rows, columns, panel_index+1))
    hex_map = ax.hexbin(low_d_projection[:, 0],
                        low_d_projection[:, 1],
                        C = closeness[:,residue_index],
                        cmap = 'YlGnBu',
                        mincnt =1,
                        vmin = 0.12,
                        vmax = 0.33,
                        gridsize = 200)
    plt.title(f'{residues[residue_index]}')
    ax.set_axis_off()
plt.savefig("EncoderMap_colored_Closeness_Centralities_MET1_ARG60.png", dpi␣
 ↪=300)
```


MET1
HIS11
THR21
LYS31
THR41
LEU51
ALA2
VAL12
PHE22
LYS32
LYS42
LEU52
PRO3
ARG13
ASP23
ILE33
VAL43
GLY53
ASN4
SER14
ALA24
LYS34
PRO44
SER54
ALA5
GLU15
ASN25
GLU35
VAL45
LYS55
SER6
GLU16
PRO26
HIS36
GLN46
ILE56
CYS7
TRP17
TYR27
VAL37
ASP47
LEU57
LEU8
ASP18
ASP28
ARG38
GLN48
LYS58
CYS9
LEU19
SER29
SER39
VAL49
PRO59
VAL10
MET20
VAL30
LYS40
LEU50
ARG60

[8]:
```
%matplotlib inline
plt.rcParams.update({'font.size': 10})
fig = plt.figure(figsize = (8,11), tight_layout = True)
rows = 10
columns = 6
counter = 1

for panel_index, residue_index in enumerate(range(60,120)):
    ax =plt.subplot(rows, columns, row_based_idx(rows, columns, panel_index+1))
    hex_map = ax.hexbin(low_d_projection[:, 0],
                        low_d_projection[:, 1],
                        C = closeness[:,residue_index],
                        cmap = 'YlGnBu',
                        mincnt =1,
                        vmin = 0.12,
                        vmax = 0.33,
                        gridsize = 200)
    plt.title(f'{residues[residue_index]}')
    ax.set_axis_off()
plt.savefig("EncoderMap_colored_Closeness_Centralities_ARG61_GLU120.png", dpi␣
 ↪=300)
```

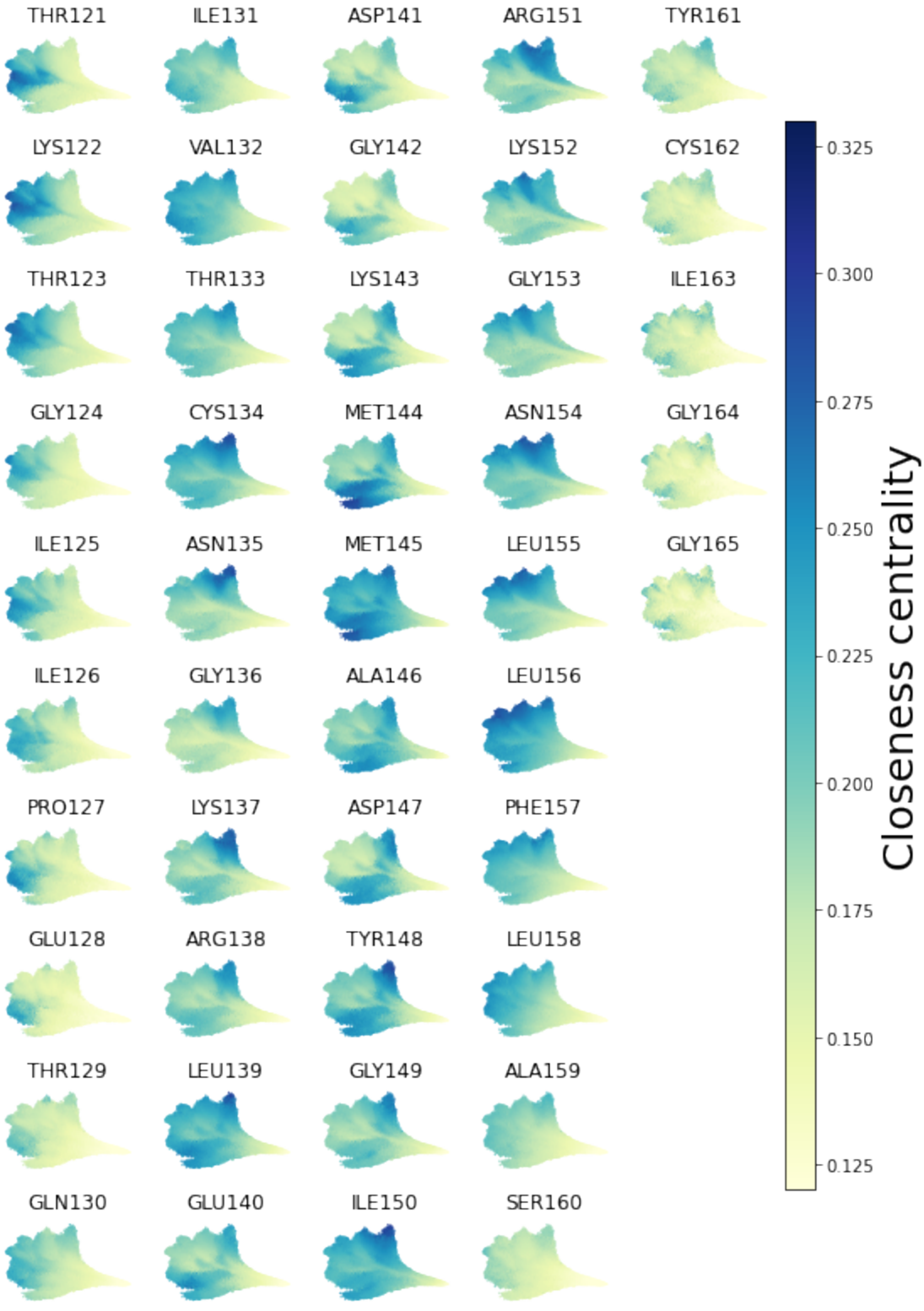

ARG61
GLU71
VAL81
PHE91
ARG101
SER111
SER62
LYS72
LYS82
LEU92
HIS102
VAL112
LEU63
THR73
PRO83
VAL93
LEU103
ALA113
SER64
ILE74
SER84
GLU94
LEU104
GLN114
SER65
HIS75
ASP85
SER95
GLN105
VAL115
TYR66
LEU76
GLU86
GLY96
VAL106
LYS116
GLY67
THR77
GLU87
ASP97
ARG107
ALA117
ILE68
LEU78
LEU88
GLU98
ARG108
MET118
ASP69
LYS79
PRO89
ALA99
SER109
ILE119
LYS70
VAL80
LEU90
LYS100
SER110
GLU120

[9]:
```
%matplotlib inline
plt.rcParams.update({'font.size': 10})
fig = plt.figure(figsize = (8,11), tight_layout = True)
rows = 10
columns = 6
counter = 1

for panel_index, residue_index in enumerate(range(120,165)):
    ax =plt.subplot(rows, columns, row_based_idx(rows, columns, panel_index+1))
    hex_map = ax.hexbin(low_d_projection[:, 0],
                        low_d_projection[:, 1],
                        C = closeness[:,residue_index],
                        cmap = 'YlGnBu',
                        mincnt =1,
                        vmin = 0.12,
                        vmax = 0.33,
                        gridsize = 200)
    plt.title(f'{residues[residue_index]}')
    ax.set_axis_off()

cbar_ax = fig.add_axes([0.83, 0.1, 0.03, 0.8])
cb = fig.colorbar(hex_map,
                  cax=cbar_ax,
                  fraction=0.1, pad=0.04)
cb.set_label('Closeness centrality', fontsize=25)
cb.set_alpha(1)
cb.draw_all()
plt.savefig("EncoderMap_colored_Closeness_Centralities_THR121_GLY165.png", dpi␣
↪=300)
```

```
<ipython-input-9-0d2ee115e893>:28: UserWarning: This figure includes Axes that
are not compatible with tight_layout, so results might be incorrect.
  plt.savefig("EncoderMap_colored_Closeness_Centralities_THR121_GLY165.png", dpi
=300)
/home/leonf/.conda/envs/Cluster_Dynamics/lib/python3.8/site-
packages/IPython/core/pylabtools.py:132: UserWarning: This figure includes Axes
that are not compatible with tight_layout, so results might be incorrect.
  fig.canvas.print_figure(bytes_io, **kw)
```


THR121
ILE131
ASP141
ARG151
TYR161
LYS122
VAL132
GLY142
LYS152
CYS162
THR123
THR133
LYS143
GLY153
ILE163
GLY124
CYS134
MET144
ASN154
GLY164
ILE125
ASN135
MET145
LEU155
GLY165
ILE126
GLY136
ALA146
LEU156
PRO127
LYS137
ASP147
PHE157
GLU128
ARG138
TYR148
LEU158
THR129
LEU139
GLY149
ALA159
GLN130
GLU140
ILE150
SER160
0.325
0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.125
Closeness centrality

0.0.7 Plot time trace for selected simuliation trajectories which display different dynamic behaviors of FAT10

- Closing (finding a state with stable domain interactions) quickly
- Closing more slowly with intermediate state
- Closing and reopening to find a different closes state
- Not closing for the duration of the simulation (200 ns)

[10]:
```
%matplotlib inline
fig = plt.figure(figsize = (16,16))
rows = 2
columns = 2
counter = 1
plt.rcParams.update({'font.size': 25})

n_frames = 2001

simulation_names = ['22 NaCl III','19 No NaCl I', '10 NaCl II', '18 NaCl III' ]


for panel_index, simulation in enumerate([146,18,84,142]):
    start = simulation * n_frames
    end = start + n_frames
    ax = plt.subplot(rows, columns, row_based_idx(rows, columns, panel_index+1))
    #ax = plt.subplots(figsize = (10,8),tight_layout = True)
    ax.scatter(low_d_projection[:, 0],
                low_d_projection[:, 1],
                s = 10,
                c = 'dimgrey',
                alpha = 1)

    traj_trace = ax.scatter(low_d_projection[start:end,0],
                            low_d_projection[start:end,1],
                            c = range(n_frames),
                            cmap= 'Blues',
                            s = 10,
                            alpha = 1)
    plt.title(f'Trajectory {simulation_names[panel_index]}')
    ax.set_axis_off()


fig.subplots_adjust(left=0.05, right=0.9, wspace=0.05)

cbar_ax = fig.add_axes([0.9, 0.15, 0.03, 0.75])
cb = fig.colorbar(traj_trace,
                  cax=cbar_ax,
                  fraction=0.1, pad=0.04,
                  label = 'Simulated time (ns)')
```

```
cb.set_alpha(1)
cb.set_ticks(range(0, n_frames, 500))
cb.set_ticklabels(['0','50','100','150','200'])
cb.draw_all()

plt.savefig("Trajectories_mapped_on_EncoderMap.png", dpi =300)
```