# 2_Plotting_arcplot_difference_graph_UMAP_clusters

June 17, 2025

## 1 Plotting difference graphs of clusters as arc diagrams

In this notebook, the difference graph of two cluster is plotted as an arc diagram.

Here, the two clusters are cluster 6 (Ile126out) and 9 (Ile126 in) in the HDBSCAN clustering of the UMAP for the C-domain.

The nodes of the consensus graph are the residues of the protein. The edges between two residues are weighted by the difference in contact probability for this edge between the clusters. Practically, this is calculated by the difference between the two mean adjacency matrices of the clusters. The edges are shown if they exceed a probability difference of 50%.

This notebook is similar to 1_Plotting_arcplot_consensus_graph_red_cluster_EncoderMap.ipynb.

## 2 Plotting python data in R using ggplot2 and ggraph.

Due to limitations in adjustability for plotting arcdiagrams in python, here the R libary `ggraph` is used for plotting. Please refer to the ggraph documentation for further information.

The data is loaded and processed using python, necessitating the use of `rpy2` as a bridge for using python and R in the same jupyter notebook. Please refer to the rpy2 documentation for further information.

### 2.1 Loading the ipython extension for rpy2

[1]:
```
%load_ext rpy2.ipython
```

### 2.2 Importing the necessary R libraries

[2]:
```
%%R
library(ggraph)
library(tidygraph)
library(repr)
library(grDevices)
library(ggplot2)
library(viridis)
```

```
Loading required package: ggplot2
```

```
Attaching package: ‘tidygraph’

The following object is masked from ‘package:stats’:

    filter

Loading required package: viridisLite
```

### 2.3 Importing the rpy2 functions to translate python objects to R objects

[3]:
```
import rpy2.robjects as ro
from rpy2.robjects import numpy2ri
from rpy2.robjects import default_converter
from rpy2.robjects.packages import importr
from rpy2.robjects import pandas2ri
```

### 2.4 Importing the necessary python packages

[4]:
```
import numpy as np
import mdtraj as md
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
```

### 2.5 This code only works if an `adjacency_matrices` array is provided!

The adjacency matrices here are loaded from disk using memory mapping in this notebook. Due to storage constraints, the adjacency matrix file is not not provided here. It was calculated in the same manner as in 1_Calculating_Closeness_Centralities_full_length_FAT10.ipynb and can be written out when running the closeness centrality calculation if needed.

[5]:
```
adjacency_matrices = np.load("/home/leonf/phd/
 ↪Analysis_FAT10_Simulation_Triplicates/
 ↪FAT10_123_adjacency_matrices_sidechain_distances_00_45_dt100_connected_bb.
 ↪npy",
                             mmap_mode = 'r')
```

### 2.6 Loading cluster ids from the HDBSCAN clustering of the UMAP

[6]:
```
cluster_ids = np.load("./2_Cluster_IDs_UMAP_HDBSCAN_isolated_domain.npy")
```

### 2.7 Defining limits of the C-domain

[7]:
```
CD_start_res = 87 # Starting residue of the C-Domain (0-Indexed. This is␣
↪residue LEU88 in the structure)
CD_end_res = 158 # End residue of the C-Domain (0-Indexed. This is ALA159 in␣
↪the structure).
```

### 2.8 Loading closeness centralities

Needed for coloring the nodes by mean closeness centrality of the cluster

[8]:
```
closeness = np.load("../Closeness_Centralities_C_domain_FAT10_res_88_159.npy")
```

### 2.9 Loading the topology for residue naming

[9]:
```
PDB_file = md.load("../start_frame_FAT10.pdb")
```

### 2.10 Creating a dictionary of nodes (i.e. residues) with residue ID and name

[10]:
```
node_dict = {index:str(residue) for index, residue in enumerate(PDB_file.
↪topology.residues) if index > CD_start_res-1 and index < CD_end_res+1}
nodes = list(node_dict.values())
```

### 2.11 Constructing pandas dataframes for nodes and edges

These are later translated into R dataframes.

[11]:
```
nodes_df = pd.DataFrame (nodes, columns = ['Residues'])
edges_df = pd.DataFrame(columns = ['from', 'to', 'weight'])
```

### 2.12 Selecting the clusters for which to construct the difference graph

[12]:
```
frames_in_cluster_0 = np.where(cluster_ids == 9)[0]
frames_in_cluster_1 = np.where(cluster_ids == 6)[0]
```

### 2.13 Calculating the difference graph

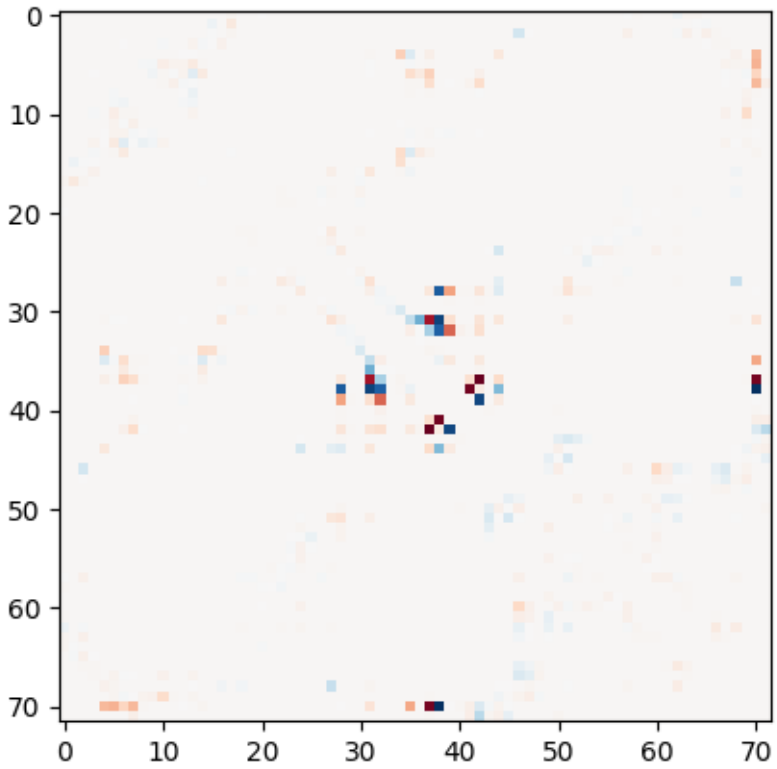
This is the difference between the two mean adjacency matrices, i.e. the difference in edge probabilities for each contact in the two clusters.

[13]:
```
adjacency_matrices_in_cluster_0 = adjacency_matrices[frames_in_cluster_0,␣
↪CD_start_res:CD_end_res+1, CD_start_res:CD_end_res+1] # CD_end_res needs to␣
↪be adjusted due to numpy array slicing syntax.
adjacency_matrices_in_cluster_1 = adjacency_matrices[frames_in_cluster_1,␣
↪CD_start_res:CD_end_res+1, CD_start_res:CD_end_res+1]
mean_adjacency_matrix_0 = np.mean(adjacency_matrices_in_cluster_0, axis = 0)
mean_adjacency_matrix_1 = np.mean(adjacency_matrices_in_cluster_1, axis = 0)
```

```
mean_difference_matrix = mean_adjacency_matrix_0-mean_adjacency_matrix_1
```

[14]:
```
plt.imshow(mean_difference_matrix, cmap = "RdBu")
```

[14]: <matplotlib.image.AxesImage at 0x7c7dac456350>



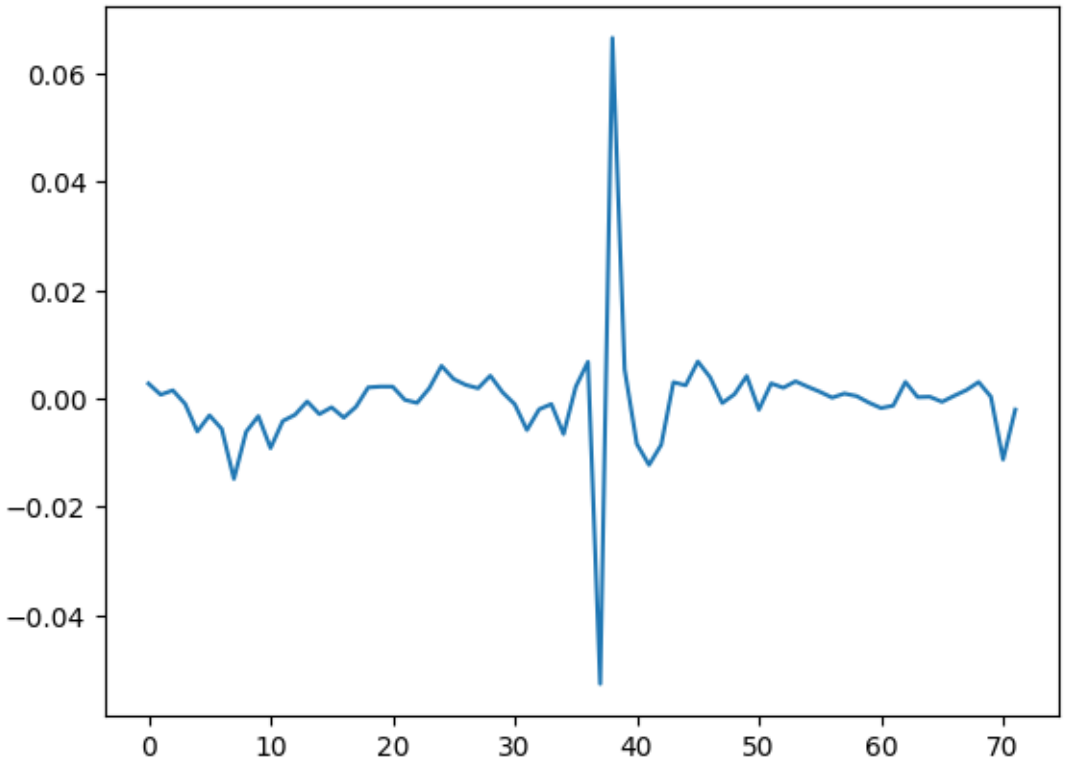

### 2.14 Adding the differnce in mean closeness centralities as a node feature

This will be used for node coloring later

[15]:
```
nodes_df['closeness'] = np.mean(closeness[frames_in_cluster_0], axis=0) - np.
↪mean(closeness[frames_in_cluster_1], axis=0)
```

[16]:
```
plt.plot(np.mean(closeness[frames_in_cluster_0], axis=0) - np.
↪mean(closeness[frames_in_cluster_1], axis=0))
```

[16]: [<matplotlib.lines.Line2D at 0x7c7dac4d1250>]




### 2.15 Setting a threshold for the minimum probability difference that should be displayed as an edge in the arc plot

[17]:
```
difference_threshold = 0.5
```

### 2.16 Adding edges to the difference graph

In this cell, we go through each residue pair in the `mean_difference_matrix`. If the probability difference of a residue-residue contact in the clusters is above the set threshold, it is added as a edge to be drawn in the arc diagram.

[18]:
```
rows =[]
# Go through each entry in the mean adjacency matrix
for i in range(mean_difference_matrix.shape[0]):
    for j in range(i +1, mean_difference_matrix.shape[1]):
        # Add edge to list of edges if probability difference for contact␣
 ↪exceeds threshold
        if np.abs(mean_difference_matrix[i,j]) > difference_threshold and␣
 ↪abs(i-j)>1:
                rows.append({"from" : int(i+1),
                             "to" : int(j+1),
                             "weight" : mean_difference_matrix[i,j]})
```

```
edges_df = pd.concat([pd.DataFrame([row]) for row in rows], ignore_index=True)
```

### 2.17 Converting pandas dataframes for nodes and edges to R dataframes using rpy2

[19]:
```
# Adjusting index from 0-Indexed (python) to 1-indexed (R)
nodes_df.index = np.arange(1, len(nodes_df)+1)
edges_df.index = np.arange(1, len(edges_df)+1)

# Converting pandas objects to R objects
with (ro.default_converter + pandas2ri.converter).context():
    r_nodes_df = ro.conversion.get_conversion().py2rpy(nodes_df)
    r_edges_df = ro.conversion.get_conversion().py2rpy(edges_df)
```

### 2.18 Plotting the arc diagram of the difference graph

[20]:
```
%%R -i r_nodes_df,r_edges_df
# Cell magic to run R in this cell and import the required objects

# construct the graph from nodes and edges
graph = tbl_graph(nodes = r_nodes_df, edges = r_edges_df)

# Plot the graph
g = ggraph(graph, layout = 'linear') +
    # Plot nodes as bars, height is given by difference in mean closeness
    geom_tile(aes(x=seq(1,72),
                  y=closeness*40,
                  height = closeness*80),
              fill = "white",
              color = 'black',
              linewidth = 0.5) +
    # Plot edges, color based on sign of the difference for clear assignment to␣
 ↪clusters
    geom_edge_arc(aes(alpha = 2,
                      color = sign(weight),
                      #width= abs(weight)  # drawing option for represeting␣
 ↪difference by width
                     ),
                  width=0.8,
                  strength = 0.9) +
    # Node labeling
    geom_node_text(aes(label = Residues),
                   angle = 90,
                   hjust = 1,
                   vjust = 0.5,
                   nudge_y = -6,
```

```
                    size = 3.5,
                    color = "black") +

    # Node plotting parameters
    scale_size_continuous(range=c(0.1,6))+
    #scale_color_viridis()+

    #Edge plotting parameters
    scale_edge_width_continuous(range=c(0.2,1))+
    scale_edge_alpha_continuous(range=c(0.1,1))+
    #Custom coloring to match cluster colors
    scale_edge_color_gradient(high= '#b8860b',
                              low = '#4682b4'
                              )+
    theme_graph() +
    theme(
    legend.position='none',
    plot.margin=unit(c(0,0,0,0), "pt"),
    panel.spacing=unit(c(0,0,0,0), "pt"),
    aspect.ratio=0.15) +
    expand_limits(x = c(-1.2, 1.2), y = c(-30,0))

# Saving the plot - displays the arcdiagram in the correct sizing and aspect␣
  ↪ratio.
ggsave(plot = g,
       width = 20,
       height = 4,
       dpi =400,
       filename ='2_Arcplot_difference_graph_UMAP_clusters.png')
g
```

```
In addition: Warning message:
Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.
  Please use `linewidth` in the `default_aes` field and elsewhere instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
generated.
```