1_Plotting_arcplot_consensus_graph_red_cluster_EncoderMap

June 17, 2025

1 Plotting consensus graphs of clusters as arc diagrams

In this notebook, the consensus graph of a cluster is plotted as an arc diagram.

Here, this is the red cluster (cluster 12) in the HDBSCAN clustering of the EncoderMap for full length FAT10.

The nodes of the consensus graph are the residues of the protein. The edges between two residues are weighted by the probability of a contact between them in a cluster. This means the adjacency matrix of the consensus graph is the mean adjacency matrix of the cluster.

Here, for readability, the edges are grouped by probability and drawn according to their probability interval:

- $[0 - 25\%[$: No edge
- $[25 - 50\%[$: dotted edge
- $[50 - 75\%[$: dashed edge
- $[75 - 100\%]$: solid edge

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')
```

[6]:
```
adjacency_matrices.shape
```

[6]:
```
(300150, 165, 165)
```

2.6 Loading cluster ids from the HDBSCAN clustering of the EncoderMap

[7]:
```
cluster_ids = np.load("./1_Cluster_IDs_Encodermap_HDBSCAN_full_length_FAT10.
↪npy")
```

2.7 Loading closeness centralities

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

[8]:
```
closeness = np.load("../Closeness_Centralities_full_length_FAT10.npy")
```

2.8 Loading the topology for residue naming

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

2.9 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)}
nodes = list(node_dict.values())
```

2.10 Selecting the cluster for which to calculate the consensus graph

[11]:
```
cluster = 12
```

2.11 Calculating the mean adjacency matrix

(i.e. the edge weights of the consensus graph)

[12]:
```
frames_in_cluster = np.where(cluster_ids == cluster)[0]
adjacency_matrices_in_cluster = adjacency_matrices[frames_in_cluster]
mean_adjacency_matrix = np.mean(adjacency_matrices_in_cluster, axis = 0)
```

2.12 Constructing pandas dataframes for nodes and edges

These are later translated into R dataframes.

[13]:
```
nodes_df = pd.DataFrame (nodes, columns = ['Residues'])
nodes_df['closeness'] = np.mean(closeness[frames_in_cluster], axis = 0)  #Node␣
↪feature: Mean closeness centrality across cluster, used for node coloring.
edges_df = pd.DataFrame(columns = ['from', 'to', 'weight'])
```

2.13 Selecting thresholds to group the edge weights of the consensus graph

[14]:
```
threshold = 0.25
threshold_dashed = 0.75
threshold_dotted = 0.5
```

2.14 Adding edges to the consensus graph

In this cell, we go through each residue pair in the mean adjacency matrix for the cluster. If the probability of a residue-residue contact in the cluster is above the set threshold, it is added as a edge to be drawn in the arc diagram.

[15]:
```
rows =[]
# Go through each entry in the mean adjacency matrix
for i in range(mean_adjacency_matrix.shape[0]):
    for j in range(i+1, mean_adjacency_matrix.shape[1]):
        # Add edge to list of edges if probability for contact exceeds␣
↪threshold and
        # residues are not direct neighbors (i.e. do not draw backbone)
        if mean_adjacency_matrix[i,j] > threshold and abs(i-j)>1:
            # ignore intra-domain contacts
            if i<85 and j>81:
                rows.append({"from" : int(i+1),
                             "to" : int(j+1),
                             # Add plotting parameters according to grouped␣
↪weight for easier visualization
                             "weight" : 1 if mean_adjacency_matrix[i,j] >=␣
↪threshold_dashed else 0.5 if mean_adjacency_matrix[i,j] >= threshold_dotted␣
↪else 0,
                             "linetype" : 1 if mean_adjacency_matrix[i,j] >=␣
↪threshold_dashed else 3 if mean_adjacency_matrix[i,j] >= threshold_dotted␣
↪else 2,
                             "curvature": 0.1})


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

2.15 Converting pandas dataframes for nodes and edges to R dataframes using rpy2

[16]:
```
# 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)
```

2.16 Plotting the arc diagram of the consensus graph

[17]:
```
%%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')+
    geom_edge_arc(aes(alpha = weight,
                      #color = weight,
                      width = weight,
                      #label = label,
                      linetype = as.factor(linetype)
                  ),
                  strength = 0.7
                  #angle_calc = "none",
                  #label_dodge = unit(-8, 'mm'),
                  #check_overlap = FALSE
                 )+

    #Plot nodes as tiles for better readability, colored by mean closeness
    geom_tile(aes(x = seq(1,165),
                  y = rep(-1.5,165),
                  fill = closeness),
                  height = 3
                   )+
    # Optional node labeling
    #geom_node_text(aes(label = Residues),
    #               angle = 90,
    #               hjust = 1,
    #               vjust = 0.5,
    #               nudge_y = -3,
    #               size = 2.5,
    #               color = "black") +
    # Node plotting parameters
    scale_fill_distiller(palette = 'YlGnBu',
                         direction = 1,
                         #trans='reverse',
                         limits=c(0.14,0.28)
                        )+
    # Edge plotting parameters
    scale_edge_width_continuous(range=c(0.4,1.8))+
    scale_edge_alpha_continuous(range=c(0.5,1))+
```

```
    scale_edge_color_distiller(palette='Spectral',
                               #trans='reverse'
                              )+
    theme_graph() +
    theme(
    legend.position='none',
    plot.margin=unit(c(-0.5,-0.5,-0.5,-0.5), "in"),
    panel.spacing=unit(c(-1,-1,-1,-1), "in"),
    #aspect.ratio=0.2
    )+
    #expand_limits(x = c(0,0), y = c(0,0))
    xlim(c(-5,170))+
    ylim(c(-10,54))

#Labeling of the domains

ymin = -7
ymax = -5
y_mean = mean(c(ymin, ymax))

g = g +
    geom_segment(aes(x = 1, xend = 165, y = y_mean, yend = y_mean), color =␣
 ↪"black", linewidth = unit(c(1.5), "pt"))+
    geom_segment(aes(x = 8, xend = 80, y = y_mean, yend = y_mean), color =␣
 ↪"steelblue", linewidth = unit(c(12), "pt"))+
    geom_segment(aes(x = 87, xend = 160, y = y_mean, yend = y_mean), color =␣
 ↪"goldenrod", linewidth = unit(c(12), "pt"))+
    annotate("text", x = mean(c(8, 80)), y = y_mean, label = "ND", size =␣
 ↪unit(c(10), "pt"), color = "black")+
    annotate("text", x = mean(c(87, 160)), y = y_mean, label = "CD", size =␣
 ↪unit(c(10), "pt"), color = "black")


# Saving the plot - displays the arcdiagram in the correct sizing and aspect␣
 ↪ratio.
ggsave(plot = g,
       width = 23,
       height = 6,
       dpi =400,
       filename ='1_Arcplot_consensus_graph_red_cluster_EncoderMap.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.
```