Appendix: Brain Hubs for Drosophila and C. Elegans

Aahana Jain

January 30, 2025

1 Basic setup of data

Using the data provided by Janelia, we organised the hemibrain connectivity into an (unweighted) adjacency matrix using the following python code:

In [2]:

```
# Adjacency matrix

import numpy as np
import pandas as pd

import networkx as nx
import matplotlib.pyplot as plt
from scipy.sparse import csr_matrix

neurons_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/traced-neurons.csv')
conn_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/traced-total-connections.csv')

# Dictionary mapping
bodyid_to_id = {bodyid: idx for idx, bodyid in enumerate(neurons_df['bodyId'])}

# Create the adj matrix
n = len(bodyid_to_id)

adj = np.zeros((n, n), dtype=np.int32)
for from_bodyid, to_bodyid in zip(conn_df['bodyId_pre'], conn_df['bodyId_post']):
    from_idx, to_idx = bodyid_to_id[from_bodyid], bodyid_to_id[to_bodyid]
    adj[from_idx, to_idx] = 1

# Row and column labels
adj_df = pd.DataFrame(adj, columns=neurons_df['bodyId'], index=neurons_df['bodyId'])

# Print the adj matrix
adj_df
```

Out[2]:

bodyId	200326126	202916528	203253072	203253253	203257652	203594169	203594175	203598499	203598504
bodyId									
200326126	0	0	0	0	0	0	0	0	0
202916528	0	0	0	1	1	0	0	0	0
203253072	0	0	0	1	0	0	1	0	1
203253253	0	0	0	0	0	1	0	1	0
203257652	0	1	0	1	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...
7112579856	0	0	0	0	0	0	0	0	0
7112615127	0	0	0	0	0	0	0	0	0
7112617294	0	0	0	0	0	0	0	0	0
7112622044	0	0	0	1	0	0	0	0	0
7112622236	0	0	0	0	0	0	0	0	0

21739 rows × 21739 columns

Here, given a simple directed graph with n vertices, the *unweighted adjacency matrix* A is defined as follows:

$$A_{ij} = \begin{cases} 1, & \text{if there is an edge from vertex } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}$$

Similarly, we found the weighted adjacency matrix using the following code:

In [1]:

```
# To find the weighted Adjacency matrix

import numpy as np
import pandas as pd

# Load the neurons and connections data
neurons_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/traced-neurons.csv')
conn_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/traced-total-connections.csv')

# Create a dictionary mapping body IDs to neuron indices
bodyid_to_id = {bodyid: idx for idx, bodyid in enumerate(neurons_df['bodyId'])}

# Create the adj matrix
n = len(bodyid_to_id)

adj = np.zeros((n, n), dtype=np.int32)  # Changed data type to float to store weights
for from_bodyid, to_bodyid, weight in zip(conn_df['bodyId_pre'], conn_df['bodyId_post'], conn_df['weight']):
    from_idx, to_idx = bodyid_to_id[from_bodyid], bodyid_to_id[to_bodyid]
    adj[from_idx, to_idx] = weight

# Set the row and column labels
adj_weight_df = pd.DataFrame(adj, columns=neurons_df['bodyId'], index=neurons_df['bodyId'])

# Print the adj matrix
adj_weight_df
```

Out[1]:

bodyId	200326126	202916528	203253072	203253253	203257652	203594169	203594175	203598499	203598504
bodyId									
200326126	0	0	0	0	0	0	0	0	0
202916528	0	0	0	2	1	0	0	0	0
203253072	0	0	0	17	0	0	1	0	1
203253253	0	0	0	0	0	2	0	7	0
203257652	0	1	0	1	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...
7112579856	0	0	0	0	0	0	0	0	0
7112615127	0	0	0	0	0	0	0	0	0
7112617294	0	0	0	0	0	0	0	0	0
7112622044	0	0	0	2	0	0	0	0	0
7112622236	0	0	0	0	0	0	0	0	0

21739 rows × 21739 columns

Here, the *weighted adjacency matrix* is defined as

$$A_{ij} = \begin{cases} w_{ij}, & \text{if there is an edge between vertex } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$$

and w_{ij} is the strength of the synaptic connection between two neurons, determined by the data in traced-total-connections.csv.

2 Calculating the Stationary Distribution

The following code was used to find the markov matrix according to the discrete time markov model:

In [1]:

```
# To find the Markov matrix

import numpy as np
import pandas as pd

# Load the neurons and connections data
neurons_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/traced-neurons.csv')
conn_df = pd.read_csv('/Users/aj/Downloads/exported-traced-adjacencies-v1.2/traced-total-connections.csv')

# (Dictionary mapping) body IDs to neuron indices
bodyid_to_idx = {bodyid: idx for idx, bodyid in enumerate(neurons_df['bodyId'])}

# Outdegree of each neuron
outdegree_dict = {bodyid: 0 for bodyid in bodyid_to_idx.keys()}
for from_bodyid in conn_df['bodyId_pre']:
    outdegree_dict[from_bodyid] += 1

# Markov matrix
n = len(bodyid_to_idx)

markov = np.zeros((n, n), dtype=np.float64)
for from_bodyid, to_bodyid in zip(conn_df['bodyId_pre'], conn_df['bodyId_post']):
    from_idx, to_idx = bodyid_to_idx[from_bodyid], bodyid_to_idx[to_bodyid]
    markov[from_idx, to_idx] = 1/outdegree_dict[from_bodyid]

# Labels
markov_df = pd.DataFrame(markov, columns=neurons_df['bodyId'], index=neurons_df['bodyId'])


markov_df
```

Out[1]:

bodyId	200326126	202916528	203253072	203253253	203257652	203594169	203594175	203598499	203598504
bodyId									
200326126	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
202916528	0.0	0.000000	0.0	0.021739	0.021739	0.000000	0.000000	0.000000	0.000000
203253072	0.0	0.000000	0.0	0.016393	0.000000	0.000000	0.016393	0.000000	0.016393
203253253	0.0	0.000000	0.0	0.000000	0.000000	0.004651	0.000000	0.004651	0.000000
203257652	0.0	0.029412	0.0	0.029412	0.000000	0.000000	0.029412	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...
7112579856	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7112615127	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7112617294	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
7112622044	0.0	0.000000	0.0	0.001908	0.000000	0.000000	0.000000	0.000000	0.000000
7112622236	0.0	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

21739 rows × 21739 columns

The stationary distribution was then found as follows:

```python
# Stationary distribution

import numpy as np
import pandas as pd
from scipy.sparse.linalg import eigs

# Create the Markov matrix
markov = np.array(markov_df)
n = len(markov)

# Eigenvectors and eigenvalues of the transpose of the Markov matrix
eigenvalues, eigenvectors = eigs(markov.T)

# Find the eigenvector corresponding to the eigenvalue closest to 1
idx = np.abs(eigenvalues - 1).argmin()
stationary = np.real(eigenvectors[:, idx].T / np.sum(eigenvectors[:, idx]))

# Convert the stationary distribution to a Pandas Series
stationary_df = pd.Series(stationary, index=markov_df.columns)

# Find the top 12 nodes with highest probabilities in the stationary distribution
top_nodes = stationary_df.nlargest(12)

# Print the top nodes
print("Nodes with highest probabilities in stationary distribution:")
print(top_nodes)
```

```
Nodes with highest probabilities in stationary distribution:
bodyId
329566174     0.000864
423101189     0.000832
425790257     0.000778
5813105172    0.000706
393766777     0.000698
518930199     0.000670
5813062858    0.000626
5813022424    0.000579
326253554     0.000566
485934965     0.000547
613079053     0.000546
668967527     0.000544
dtype: float64
```

Similarly, we get the following stationary distribution for C. Elegans (note that the graph here is unweighted):

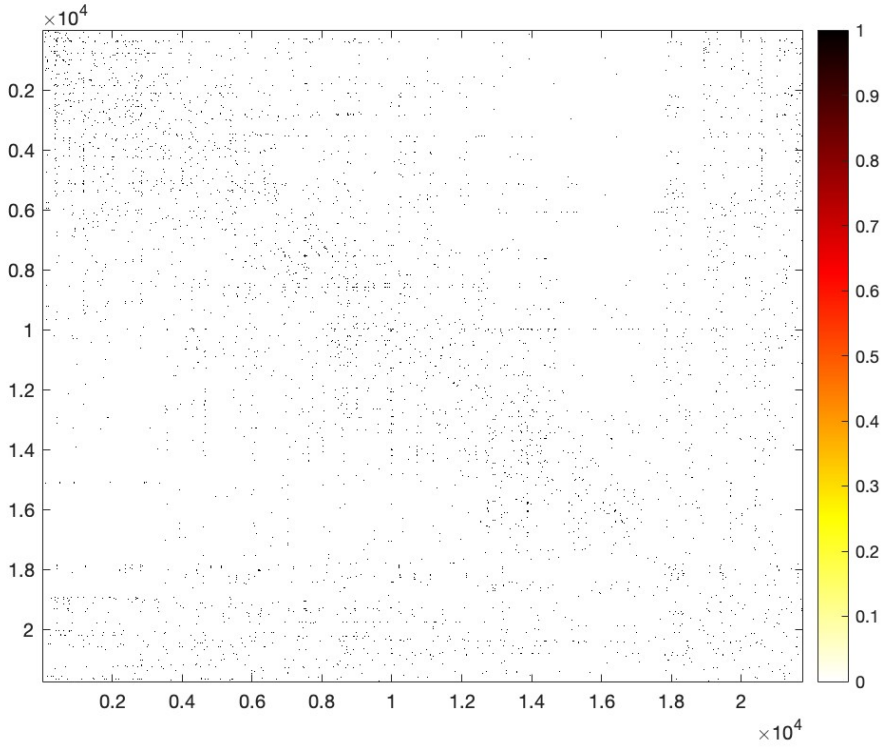
```
Nodes with highest probabilities in stationary distribution:
150    5.000000e-01
151    5.000000e-01
218    3.449431e-17
243    5.333811e-18
98     4.545971e-18
95     3.602453e-18
33     3.036580e-18
188    2.966766e-18
169    2.663199e-18
87     2.609910e-18
26     2.571551e-18
248    2.071972e-18
dtype: float64
```

3 Heatmap corresponding to the adjacency matrix

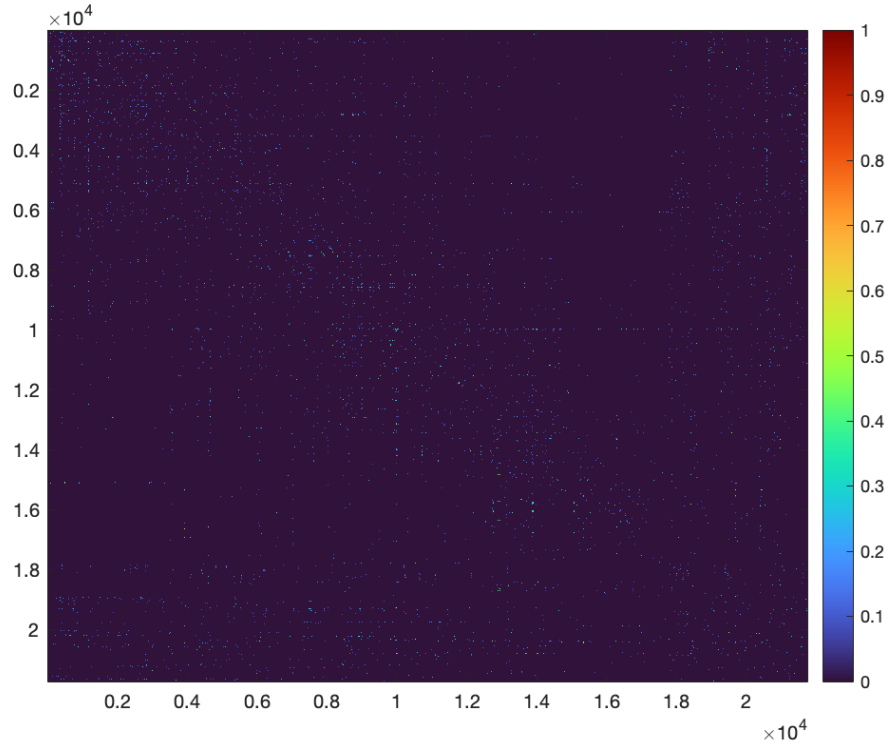
We used the following command in MATLAB to generate the heatmaps corresponding to a given adjacency matrix:

```
try
    adjacencyMatrix = readmatrix('Users/aj/Desktop/Brain Hubs/adjacency.csv');
catch
    error('Error: adjacency.csv not found. Ensure the file is in the current directory.');
end

adjacency = adjacencyMatrix(2:21740, 1:21739);

% Use the following code to scale the weighted adjacency matrix for clearer
% visualisation
% logMatrix = log1p(adjacency); % log scaling
% logMatrix = logMatrix / max(logMatrix(:)); % normalising

figure;
colormap(flipud(hot))
imagesc(adjacency);
colorbar;
```

Heatmap corresponding to the unweighted adjacency matrix for the *Drosophila* hemibrain connectome:



Heatmap corresponding to the weighted adjacency matrix for the *Drosophila* hemibrain connectome:



Heatmap corresponding to the unweighted adjacency matrix for the *C. Elegans* connectome: