Analyzing the Protein-Protein Interaction Network

Fall 22, SBE3031 - Advanced Topics in Medical Informatics

Ibrahim Mohamed Youssef, PhD

Assistant Professor

Abdelrahman Ali Farouk

Mahmoud Yaser Salman

Nevein Mohamed Ayman

Omar Saad EL-Gharbawy

February 6, 2023

Table of Contents

- Introduction

The protein-protein interaction network plays a critical role in various biological processes, such as signaling pathways and networks. Protein–protein interactions (PPIs) are physical contacts of high specificity established between two or more protein molecules, proteins do not function in isolation; instead, it is their interactions with one another and with other molecules that mediate metabolic and signaling pathways, cellular processes, and organismal systems, protein interaction data is incredibly important. It describes the interplay between the biomolecules encoded by genes. It allows us to understand the complexities of cellular function and even predict potential therapeutics. In this research paper, we will focus on analyzing these interactions. We will utilize computational techniques to map and study the network, providing valuable insights into the organization and regulation of these interactions.

The interactome used in this study represents a directed network of PPIs and was obtained from the PathLinker database. Using the NetworkX python package, we will construct a graph from the interactome and perform various analyses on the network, including finding the shortest path between two proteins, identifying directly connected proteins, ranking proteins based on their degree, converting UniProt IDs to gene names, and converting the graph to an unweighted graph.

- Methods

Used Packages:

1- **NetworkX** is a powerful Python package used to create, manipulate, and analyze complex network structures. It provides a variety of algorithms and data structures to support the efficient handling of graph-based data, making it an ideal tool for the study of protein-protein interactions.

2- **UniProt** is a freely accessible database of protein sequence and functional information, many entries being derived from genome sequencing projects. It is maintained by the UniProt consortium, which consists of several European bioinformatics organizations and a foundation. The proteome identifier (UPID) is the unique identifier assigned to the protein set that constitutes the proteome. It consists of the characters 'UP' followed by 9 digits, is stable across releases, and can therefore be used to cite a UniProt proteome.

3- **Pandas** is Python's widely used data manipulation and analysis library. It provides fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. We used it for data exploration.

4- **NumPy** is a powerful library for scientific computing in Python. It provides support for arrays, which are essential for numerical computing and data analysis, as well as mathematical functions and tools to manipulate arrays efficiently. We used it in constructing an adjacency matrix.

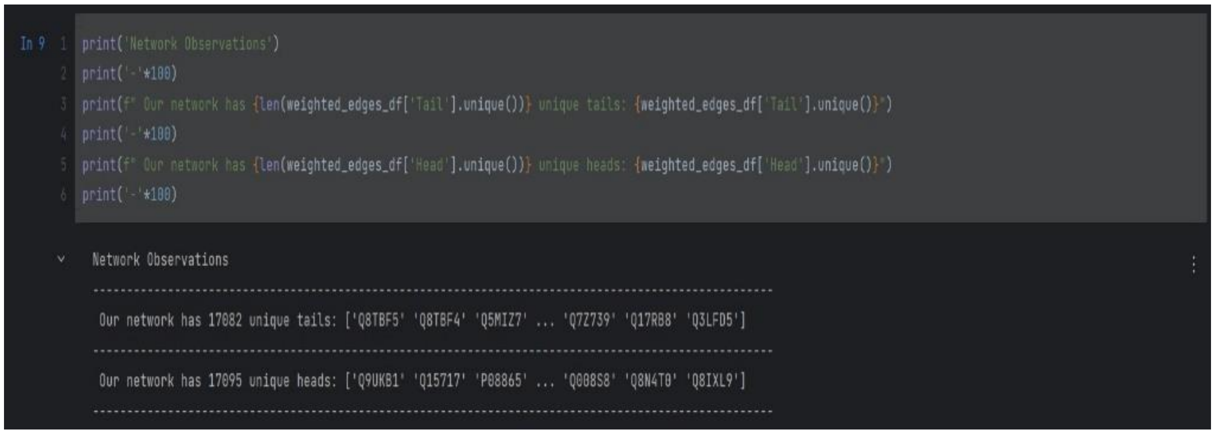
5- **Matplotlib and Seaborn** are widely used Python data visualization libraries that allow for creating interactive, publication-quality graphs and charts. We used it for visualizing all our figures.

6- **StringIO** from the **io** library allows for the creation of a virtual file-like object from a string. We used it for reading **UniProt** gene information from a given protein.

Steps:

- **Reading Text File Data.**
    - Unzipped the data file and read the data using python.
    - Extracting proteins' tails, heads, and edge weights.

- **Data Exploration with Pandas.**



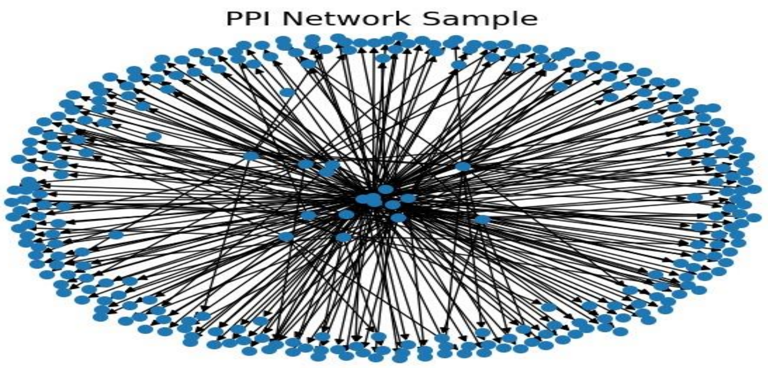
```
In 9
print('Network Observations')
print('-'*100)
print(f" Our network has {len(weighted_edges_df['Tail'].unique())} unique tails: {weighted_edges_df['Tail'].unique()}")
print('-'*100)
print(f" Our network has {len(weighted_edges_df['Head'].unique())} unique heads: {weighted_edges_df['Head'].unique()}")
print('-'*100)
```

```
Network Observations
----------------------------------------------------------------------------------------------------
 Our network has 17082 unique tails: ['Q8TBF5' 'Q8TBF4' 'Q5MIZ7' ... 'Q7Z739' 'Q17RB8' 'Q3LFD5']
----------------------------------------------------------------------------------------------------
 Our network has 17095 unique heads: ['Q9UKB1' 'Q15717' 'P08865' ... 'Q008S8' 'Q8N4T0' 'Q8IXL9']
----------------------------------------------------------------------------------------------------
```

- As the number of heads is very close to the number of tails, we may assume that the network is balanced. and this assumption can lead to several insights:
    - Network stability: A balanced PPI network is likely to be more stable and less prone to disruption than an imbalanced network.
    - Symmetrical interactions: The equal number of unique tails and heads suggests that the interactions in the network are symmetrical and that proteins are likely to be interacting with each other in a reciprocal manner.
    - Functionality: Proteins in a balanced PPI network are likely to be playing *similar roles* in the network, regardless of their specific interactions.

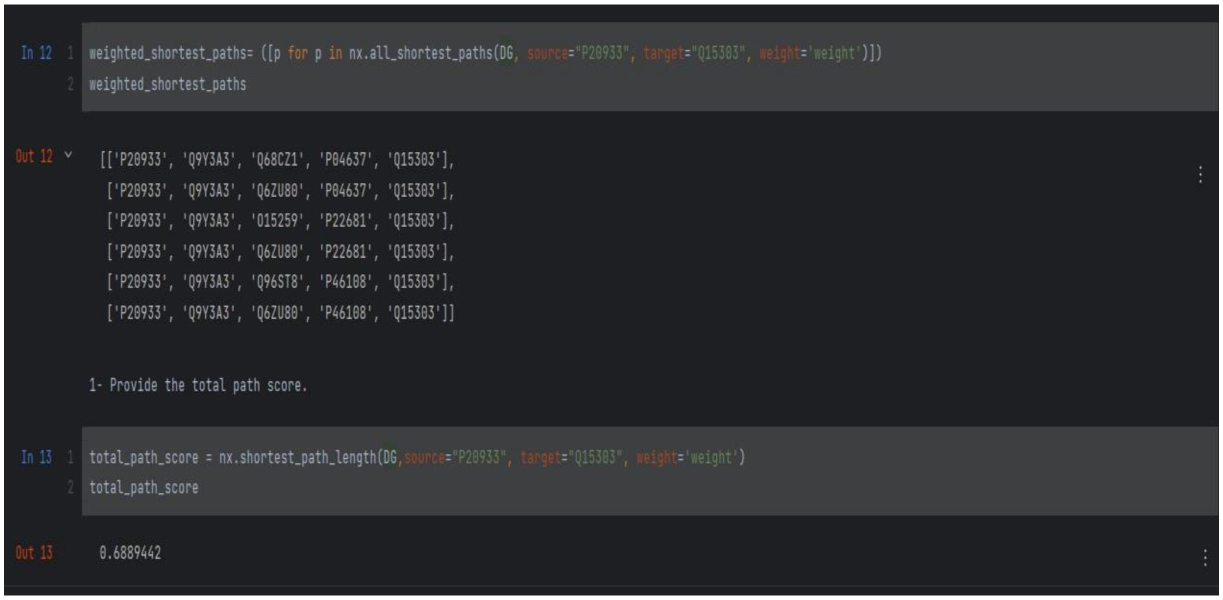
- Lack of dominant hub proteins: If a network is balanced, it is less likely to have a *small number* of highly connected hub proteins that dominate the network.
- Redundancy: A balanced PPI network is likely to have a high degree of redundancy, with many proteins having similar interactions and functional roles.

- **Construct a graph (biological network) from the above interactome.**



- By using NetworkX package we constructed a graph by taking a sample of 300 edge out of 612,516

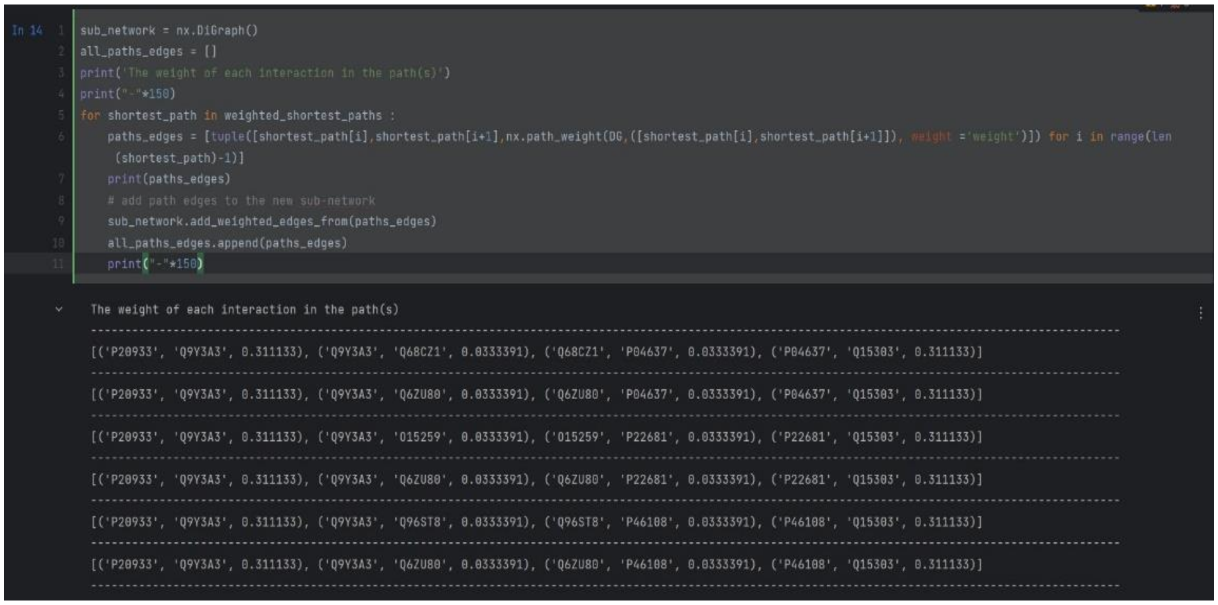
- **Given two proteins, list the acyclic shortest path(s) between these two nodes in a text file.**



```
In 12
weighted_shortest_paths= ([p for p in nx.all_shortest_paths(DG, source="P20933", target="Q15303", weight='weight')])
weighted_shortest_paths

Out 12
 [['P20933', 'Q9Y3A3', 'Q68CZ1', 'P04637', 'Q15303'],
  ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P04637', 'Q15303'],
  ['P20933', 'Q9Y3A3', 'O15259', 'P22681', 'Q15303'],
  ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P22681', 'Q15303'],
  ['P20933', 'Q9Y3A3', 'Q96ST8', 'P46108', 'Q15303'],
  ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P46108', 'Q15303']]

1- Provide the total path score.

In 13
total_path_score = nx.shortest_path_length(DG,source="P20933", target="Q15303", weight='weight')
total_path_score

Out 13
 0.6889442
```

- Provide the total path score.
  - Weighted graphs:
    - Weights are costs: the path with the minimum total weight of its edges (the minimum cost), and it is the path with the minimum energy consumption in the cell.
    - Weights are probabilities of interaction: the path with the maximum total weight of its edges (the most probable path), and it is the path with the highest interaction rate.
    - So, we conclude that the shortest path in our case is the path that is most likely to happen. Assuming that the edge weights are cost weights.

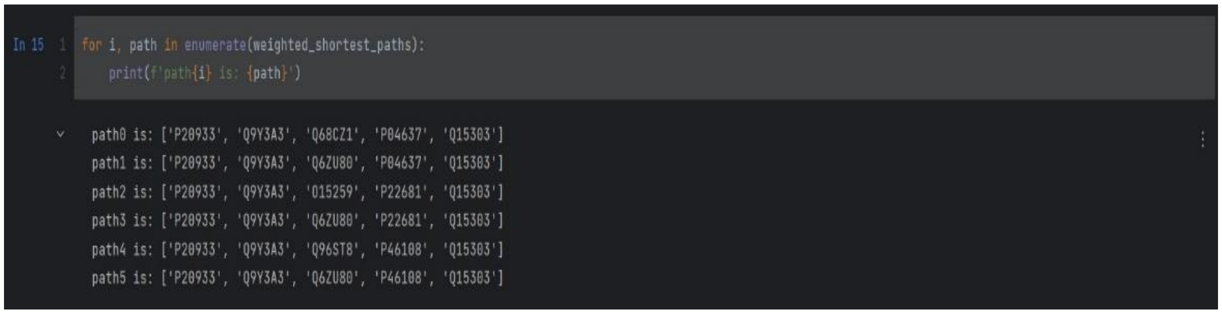
- Provide the weight of each interaction in the path(s).



```
In 14
sub_network = nx.DiGraph()
all_paths_edges = []
print('The weight of each interaction in the path(s)')
print("-"*150)
for shortest_path in weighted_shortest_paths :
    paths_edges = [tuple([shortest_path[i],shortest_path[i+1],nx.path_weight(DG,([shortest_path[i],shortest_path[i+1]]), weight ='weight')]) for i in range(len
     (shortest_path)-1)]
    print(paths_edges)
    # add path edges to the new sub-network
    sub_network.add_weighted_edges_from(paths_edges)
    all_paths_edges.append(paths_edges)
    print("-"*150)
```

```
The weight of each interaction in the path(s)
------------------------------------------------------------------------------------------------------------------------------------------------------
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q68CZ1', 0.0333391), ('Q68CZ1', 'P04637', 0.0333391), ('P04637', 'Q15303', 0.311133)]
------------------------------------------------------------------------------------------------------------------------------------------------------
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q6ZU80', 0.0333391), ('Q6ZU80', 'P04637', 0.0333391), ('P04637', 'Q15303', 0.311133)]
------------------------------------------------------------------------------------------------------------------------------------------------------
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'O15259', 0.0333391), ('O15259', 'P22681', 0.0333391), ('P22681', 'Q15303', 0.311133)]
------------------------------------------------------------------------------------------------------------------------------------------------------
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q6ZU80', 0.0333391), ('Q6ZU80', 'P22681', 0.0333391), ('P22681', 'Q15303', 0.311133)]
------------------------------------------------------------------------------------------------------------------------------------------------------
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q96ST8', 0.0333391), ('Q96ST8', 'P46108', 0.0333391), ('P46108', 'Q15303', 0.311133)]
------------------------------------------------------------------------------------------------------------------------------------------------------
[('P20933', 'Q9Y3A3', 0.311133), ('Q9Y3A3', 'Q6ZU80', 0.0333391), ('Q6ZU80', 'P46108', 0.0333391), ('P46108', 'Q15303', 0.311133)]
------------------------------------------------------------------------------------------------------------------------------------------------------
```

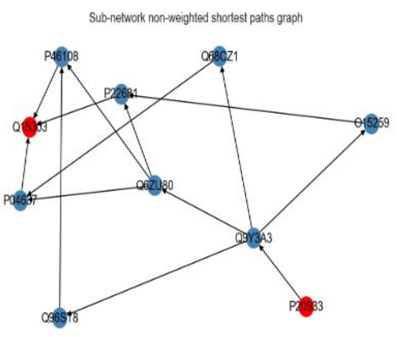
- If more than one path, report all the paths.

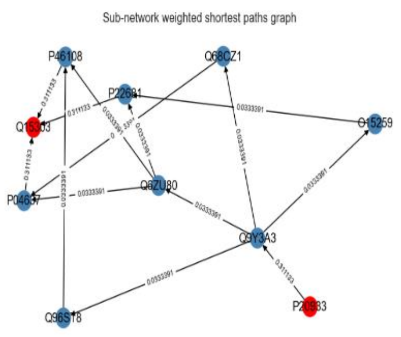


```
In 15
for i, path in enumerate(weighted_shortest_paths):
    print(f'path{i} is: {path}')
```

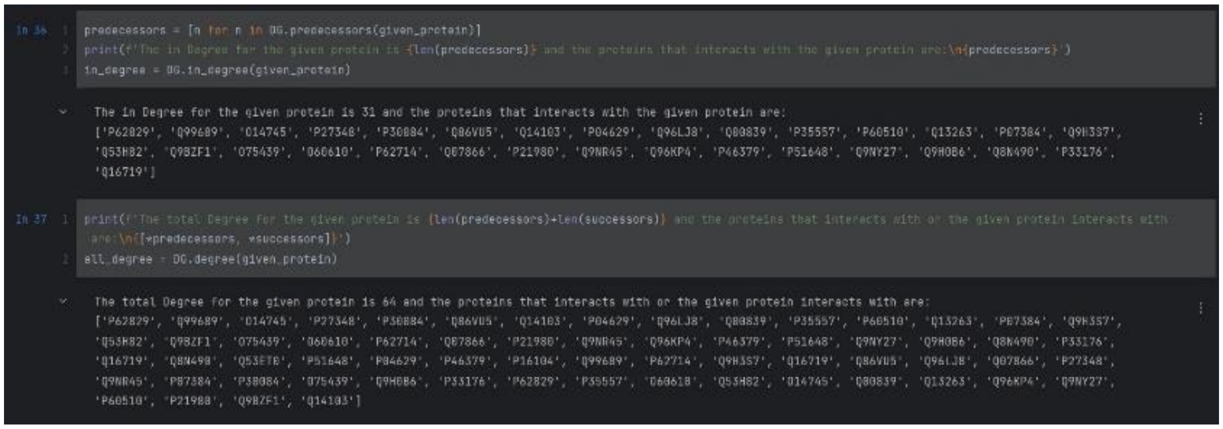
```
path0 is: ['P20933', 'Q9Y3A3', 'Q68CZ1', 'P04637', 'Q15303']
path1 is: ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P04637', 'Q15303']
path2 is: ['P20933', 'Q9Y3A3', 'O15259', 'P22681', 'Q15303']
path3 is: ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P22681', 'Q15303']
path4 is: ['P20933', 'Q9Y3A3', 'Q96ST8', 'P46108', 'Q15303']
path5 is: ['P20933', 'Q9Y3A3', 'Q6ZU80', 'P46108', 'Q15303']
```

- Use NetworkX and matplotlib to draw the sub-network formed by these shortest paths.





- **Given one protein, list all the directly connected proteins to it in a text file.**
  - Report the degree (number of connections) of this protein in a separate line.



```
In 36
predecessors = [n for n in DG.predecessors(given_protein)]
print(f'The in Degree for the given protein is {len(predecessors)} and the proteins that interacts with the given protein are:\n{predecessors}')
in_degree = DG.in_degree(given_protein)
```

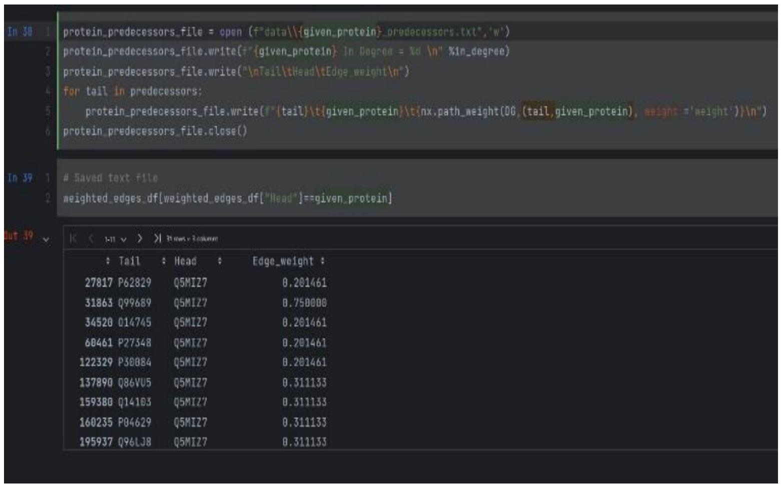
```
The in Degree for the given protein is 31 and the proteins that interacts with the given protein are:
['P62829', 'Q99689', 'O14745', 'P27348', 'P30084', 'Q86VU5', 'Q14103', 'P04629', 'Q96LJ8', 'Q80839', 'P35557', 'P60510', 'Q13263', 'P07384', 'Q9H3S7',
'Q53H82', 'Q9BZF1', 'O75439', 'O60610', 'P62714', 'Q07866', 'P21980', 'Q9NR45', 'Q96KP4', 'P46379', 'P51648', 'Q9NY27', 'Q9H0B6', 'Q8N490', 'P33176',
'Q16719']
```

```
In 37
print(f'The total Degree for the given protein is {len(predecessors)+len(successors)} and the proteins that interacts with or the given protein interacts with
 are:\n{[*predecessors, *successors]}')
all_degree = DG.degree(given_protein)
```

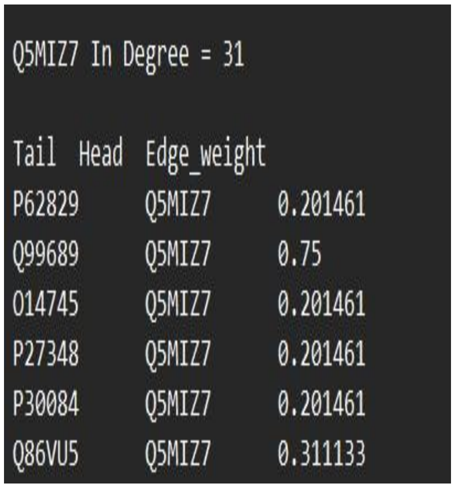
```
The total Degree for the given protein is 64 and the proteins that interacts with or the given protein interacts with are:
['P62829', 'Q99689', 'O14745', 'P27348', 'P30084', 'Q86VU5', 'Q14103', 'P04629', 'Q96LJ8', 'Q80839', 'P35557', 'P60510', 'Q13263', 'P07384', 'Q9H3S7',
'Q53H82', 'Q9BZF1', 'O75439', 'O60610', 'P62714', 'Q07866', 'P21980', 'Q9NR45', 'Q96KP4', 'P46379', 'P51648', 'Q9NY27', 'Q9H0B6', 'Q8N490', 'P33176',
'Q16719', 'Q8N490', 'Q53ET0', 'P51648', 'P04629', 'P46379', 'P16104', 'Q99689', 'P62714', 'Q9H3S7', 'Q16719', 'Q86VU5', 'Q96LJ8', 'Q07866', 'P27348',
'Q9NR45', 'P07384', 'P30084', 'O75439', 'Q9H0B6', 'P33176', 'P62829', 'P35557', 'O60610', 'Q53H82', 'O14745', 'Q80839', 'Q13263', 'Q96KP4', 'Q9NY27',
'P60510', 'P21980', 'Q9BZF1', 'Q14103']
```

- **Conclusion**: when in-degree is close to out-degree that can result in stability of overall network topological, thus some studies have suggested that a more stable network may decrease the probability of disease infection by providing more robust connections between proteins, reducing the risk of network failure and disease spread. However, this is not a universal trend, and stability in the network topology may not always result in decreased disease infection. Further research is needed to fully understand the relationship between network stability, network topology, and disease infection.

- Provide each connected protein in a line with its corresponding interaction weight.

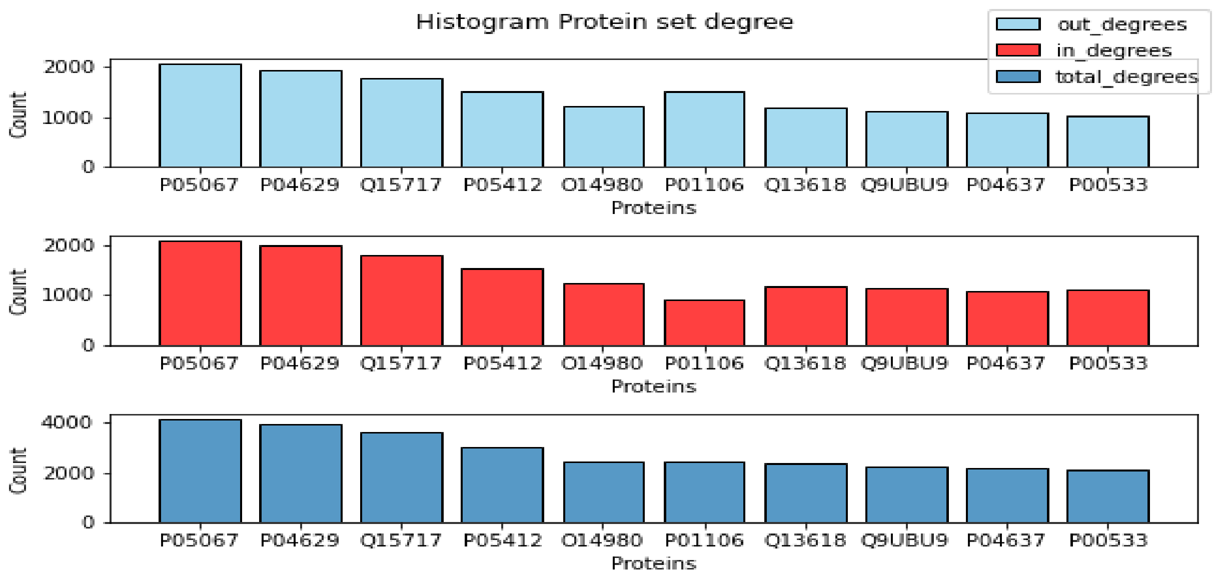


```
In 38
protein_predecessors_file = open (f"data\\{given_protein}_predecessors.txt",'w')
protein_predecessors_file.write(f"{given_protein} In Degree = %d \n" %in_degree)
protein_predecessors_file.write("\nTail\tHead\tEdge_weight\n")
for tail in predecessors:
    protein_predecessors_file.write(f"{tail}\t{given_protein}\t{nx.path_weight(DG,(tail,given_protein), weight ='weight')}\n")
protein_predecessors_file.close()

In 39
# Saved text file
weighted_edges_df[weighted_edges_df["Head"]==given_protein]

Out 39
          Tail      Head        Edge_weight
 27817  P62829    Q5MIZ7           0.201461
 31863  Q99689    Q5MIZ7           0.750000
 34520  O14745    Q5MIZ7           0.201461
 60461  P27348    Q5MIZ7           0.201461
122329  P30084    Q5MIZ7           0.201461
137890  Q86VU5    Q5MIZ7           0.311133
159380  Q14103    Q5MIZ7           0.311133
160235  P04629    Q5MIZ7           0.311133
195937  Q96LJ8    Q5MIZ7           0.311133
```



```
Q5MIZ7 In Degree = 31

Tail    Head    Edge_weight
P62829  Q5MIZ7  0.201461
Q99689  Q5MIZ7  0.75
O14745  Q5MIZ7  0.201461
P27348  Q5MIZ7  0.201461
P30084  Q5MIZ7  0.201461
Q86VU5  Q5MIZ7  0.311133
```

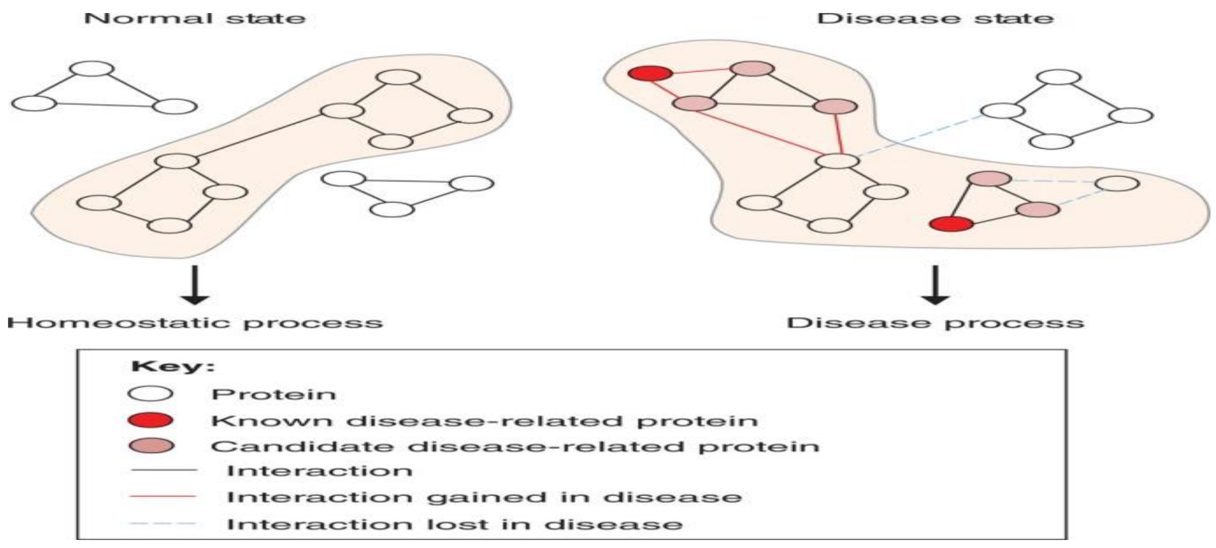
- **Given a set of proteins**:
  - Draw a histogram for the protein's degree.



  - We can conclude that both `P05067` and `P04629` are hub proteins and may be key scaffold proteins.
  - We notice that P05067 & P04629 are hub proteins and may be key scaffold proteins (based on its protein complex), and as hub proteins play a central role in the organization and function of the network. The importance of hub proteins can be attributed to several factors, including:

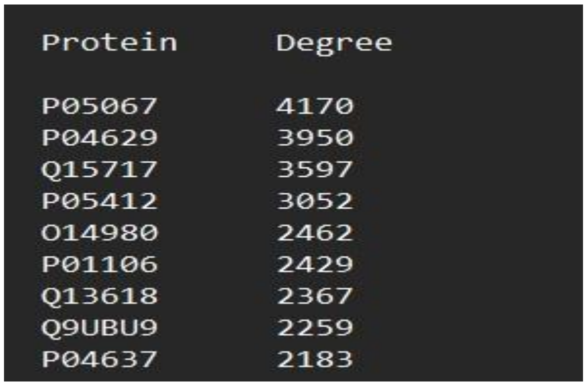
- Modularity: Hub proteins can act as central connectors between different modules or functional units in the network, allowing for effective communication and coordination of cellular processes.
- Robustness: Hub proteins can provide stability to the network by forming multiple interactions with other proteins, making the network less susceptible to failure or perturbations.
- Functionality: Hub proteins are often involved in multiple pathways and functions, making them crucial for the overall function of the cell.
- Evolution: Hub proteins tend to evolve slowly and are conserved across species, suggesting their importance in the evolution and stability of cellular processes.
- Disease: Disruptions in the interactions of hub proteins can have a significant impact on cellular processes and have been implicated in various diseases, such as cancer.

- Diseases of interactions(PPI)
  - Many human diseases are the result of abnormal protein–protein interactions involving endogenous proteins, proteins from pathogens or both. The inhibition of these aberrant associations is of obvious clinical significance. Because of the diverse nature of protein–protein interactions, however, the successful design of therapeutics requires detailed knowledge of each system at a molecular and atomic level and new undesired protein interactions are the main causes of several diseases.
  - View of how **disease-related proteins** can drive disease processes by altering individual protein complexes and protein network dynamics.


Normal state
Disease state
Homeostatic process
Disease process
Key:
Protein
Known disease-related protein
Candidate disease-related protein
Interaction
Interaction gained in disease
Interaction lost in disease

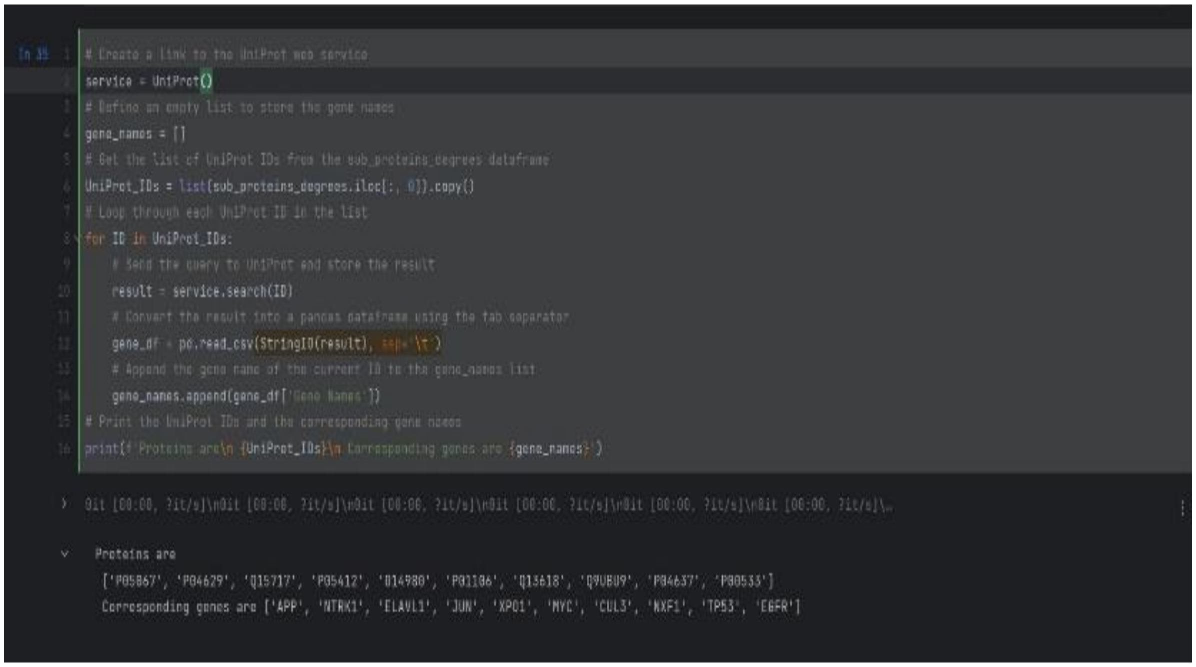
- So, our subnetwork is robust, modular, & stable. and their hub proteins have some characteristics:
    - P05067
        - **Function**
            - Functions as a cell surface receptor and performs physiological functions on the surface of neurons relevant to neurite growth, neuronal adhesion and axonogenesis. Interaction between APP molecules on neighboring cells promotes synaptogenesis Involved in cell mobility and transcription regulation through protein-protein interactions. Can promote transcription activation through binding to APBB1-KAT5 and inhibits Notch signaling through interaction with Numb. Couples to apoptosis-inducing pathways such as those mediated by G(o) and JIP. Inhibits G(o) alpha ATPase activity (By similarity).
        - **Disease**
            - Alzheimer’s disease 1 (AD1)
                - The disease is caused by variants affecting the gene represented in this entry.
            - Cerebral amyloid angiopathy, APP-related (CAA-APP)
                - The disease is caused by variants affecting the gene represented in this entry.
    - P04629
        - **Function**
            - Receptor tyrosine kinase is involved in the development and maturation of the central and peripheral nervous systems through the regulation of proliferation, differentiation, and survival of sympathetic and nervous neurons. High-affinity receptor for NGF which is its primary.
        - **Disease**
            - Congenital insensitivity to pain with anhidrosis (CIPA). The disease is caused by variants affecting the gene represented in this entry Characterized by congenital insensitivity to pain, anhidrosis (absence of sweating), absence of reaction to noxious stimuli, self-mutilating behavior, and intellectual disability.

- Rank these proteins from the highly connected to the least in a text file, where each line is a protein and its corresponding degree.



```
Protein      Degree

P05067       4170
P04629       3950
Q15717       3597
P05412       3052
O14980       2462
P01106       2429
Q13618       2367
Q9UBU9       2259
P04637       2183
```

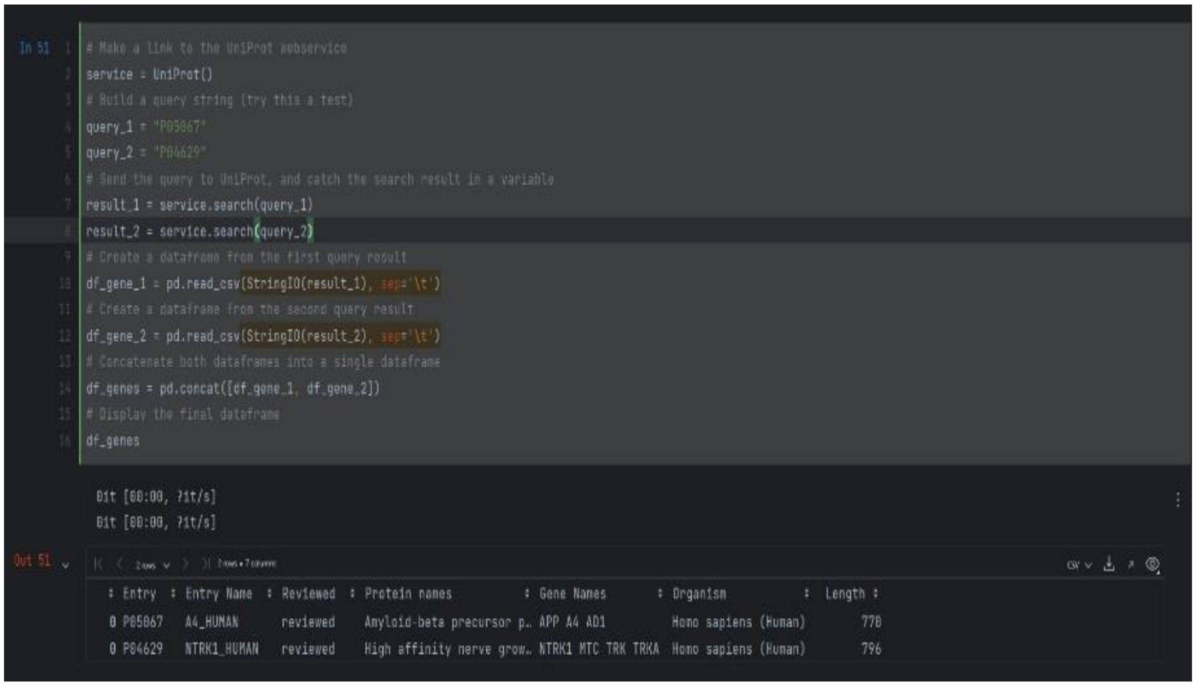
- **Provide a conversion map between the protein UniProt ID and its gene name.**
  - You can be provided with one protein ID or a set of protein IDs, and then you need to get their corresponding gene names.



```
In 35
# Create a link to the UniProt web service
service = UniProt()
# Define an empty list to store the gene names
gene_names = []
# Get the list of UniProt IDs from the sub_proteins_degrees dataframe
UniProt_IDs = list(sub_proteins_degrees.iloc[:, 0]).copy()
# Loop through each UniProt ID in the list
for ID in UniProt_IDs:
    # Send the query to UniProt and store the result
    result = service.search(ID)
    # Convert the result into a pandas dataframe using the tab separator
    gene_df = pd.read_csv(StringIO(result), sep='\t')
    # Append the gene name of the current ID to the gene_names list
    gene_names.append(gene_df['Gene Names'])
# Print the UniProt IDs and the corresponding gene names
print(f'Proteins are\n {UniProt_IDs}\n Corresponding genes are {gene_names}')
```

```
0it [00:00, ?it/s]\n0it [00:00, ?it/s]\n0it [00:00, ?it/s]\n0it [00:00, ?it/s]\n0it [00:00, ?it/s]\n0it [00:00, ?it/s]\…

 Proteins are
 ['P05067', 'P04629', 'Q15717', 'P05412', 'O14980', 'P01106', 'Q13618', 'Q9UBU9', 'P04637', 'P00533']
 Corresponding genes are ['APP', 'NTRK1', 'ELAVL1', 'JUN', 'XPO1', 'MYC', 'CUL3', 'NXF1', 'TP53', 'EGFR']
```

- As we concluded before that we have two hub proteins, in this section we retrieved their corresponding genes.



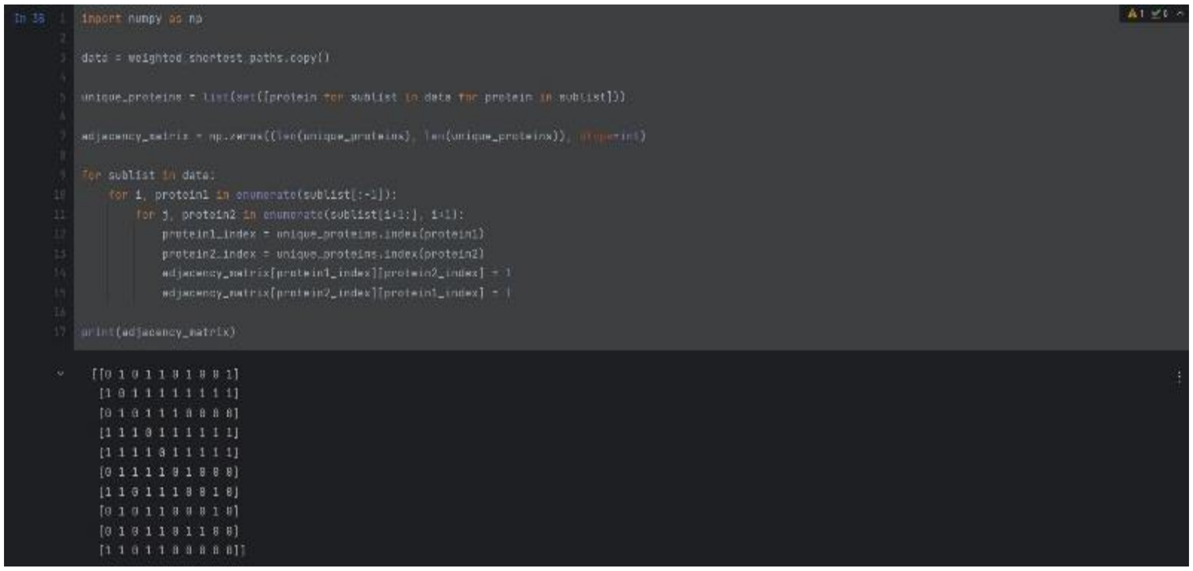
```
In 51
# Make a link to the UniProt webservice
service = UniProt()
# Build a query string (try this a test)
query_1 = "P05067"
query_2 = "P04629"
# Send the query to UniProt, and catch the search result in a variable
result_1 = service.search(query_1)
result_2 = service.search(query_2)
# Create a dataframe from the first query result
df_gene_1 = pd.read_csv(StringIO(result_1), sep='\t')
# Create a dataframe from the second query result
df_gene_2 = pd.read_csv(StringIO(result_2), sep='\t')
# Concatenate both dataframes into a single dataframe
df_genes = pd.concat([df_gene_1, df_gene_2])
# Display the final dataframe
df_genes
```

```
0it [00:00, ?it/s]
0it [00:00, ?it/s]
```

Out 51

	Entry	Entry Name	Reviewed	Protein names	Gene Names	Organism	Length
0	P05067	A4_HUMAN	reviewed	Amyloid-beta precursor p…	APP A4 AD1	Homo sapiens (Human)	770
0	P04629	NTRK1_HUMAN	reviewed	High affinity nerve grow…	NTRK1 MTC TRK TRKA	Homo sapiens (Human)	796

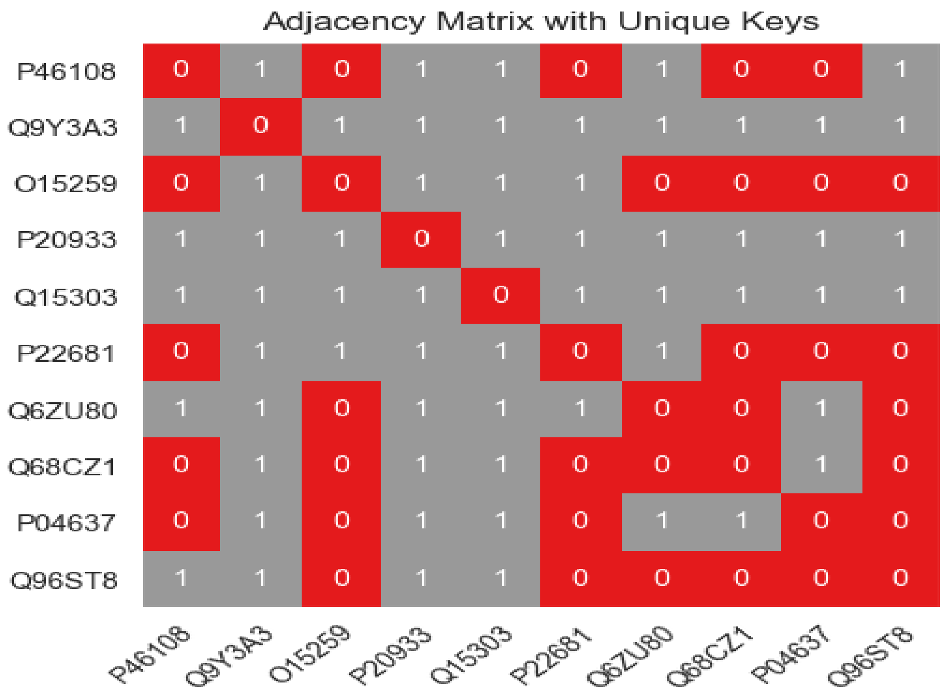
- **Convert the above graph as an unweighted graph and save it using the adjacency matrix method.**



```
In 36
import numpy as np

data = weighted_shortest_paths.copy()

unique_proteins = list(set([protein for sublist in data for protein in sublist]))

adjacency_matrix = np.zeros((len(unique_proteins), len(unique_proteins)), dtype=int)

for sublist in data:
    for i, protein1 in enumerate(sublist[:-1]):
        for j, protein2 in enumerate(sublist[i+1:], i+1):
            protein1_index = unique_proteins.index(protein1)
            protein2_index = unique_proteins.index(protein2)
            adjacency_matrix[protein1_index][protein2_index] = 1
            adjacency_matrix[protein2_index][protein1_index] = 1

print(adjacency_matrix)
```

```
[[0 1 0 1 1 0 1 0 0 1]
 [1 0 1 1 1 1 1 1 1 1]
 [0 1 0 1 1 1 0 0 0 0]
 [1 1 1 0 1 1 1 1 1 1]
 [1 1 1 1 0 1 1 1 1 1]
 [0 1 1 1 1 0 1 0 0 0]
 [1 1 0 1 1 1 0 0 1 0]
 [0 1 0 1 1 0 0 0 1 0]
 [0 1 0 1 1 0 1 1 0 0]
 [1 1 0 1 1 0 0 0 0 0]]
```

- We constructed this adjacency matrix from the last network we created of shortest paths where "P20933" is the tail, "Q15303" is the head, and "0.6889442" is the weight cost.



Adjacency Matrix with Unique Keys

	P46108	Q9Y3A3	O15259	P20933	Q15303	P22681	Q6ZU80	Q68CZ1	P04637	Q96ST8
P46108	0	1	0	1	1	0	1	0	0	1
Q9Y3A3	1	0	1	1	1	1	1	1	1	1
O15259	0	1	0	1	1	1	0	0	0	0
P20933	1	1	1	0	1	1	1	1	1	1
Q15303	1	1	1	1	0	1	1	1	1	1
P22681	0	1	1	1	1	0	1	0	0	0
Q6ZU80	1	1	0	1	1	1	0	0	1	0
Q68CZ1	0	1	0	1	1	0	0	0	1	0
P04637	0	1	0	1	1	0	1	1	0	0
Q96ST8	1	1	0	1	1	0	0	0	0	0

- **Q9Y3A3** is the hub protein of this sub-network.

- Conclusion

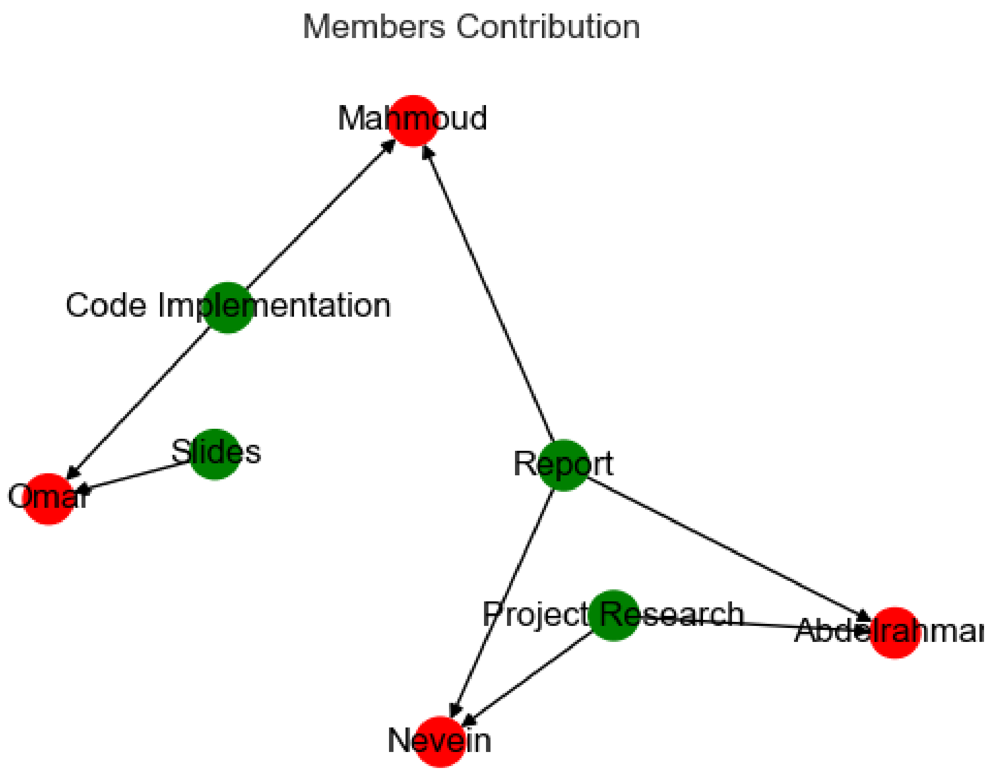
In conclusion, the analysis of protein-protein interactions is a critical step toward gaining a comprehensive understanding of biological systems and the formation of complex biological networks. This research paper utilized computational techniques to analyze the protein-protein interaction network, including utilizing the NetworkX python package, and UniProt database, and visualizing the results with matplotlib and seaborn. The tasks performed in this research paper provided valuable insights into the organization and regulation of protein-protein interactions, including finding the shortest path between two proteins, identifying directly connected proteins, ranking proteins based on their degree, converting UniProt IDs to gene names, and converting the graph to an unweighted graph. This study is a testament to the power of computational analysis in understanding the complexities of cellular function and predicting potential therapeutics.

- References

- *Degree distribution. indegree and outdegree*. (2011, October). Retrieved February 2023, from https://www.researchgate.net/figure/Degree-Distribution-Indegree-and-outdegree-distribution-averaged-over-5-time-points-The_fig2_51740396
- Protein-Protein Interaction(PPI). (2012, December). Retrieved February 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3531279/
- Biological network and interactions effects. (2013, April). Retrieved February 2023, from https://genomemedicine.biomedcentral.com/articles/10.1186/gm441
- Using graph theory to analyze biological networks. (2011, April). Retrieved February 2023, from https://biodatamining.biomedcentral.com/articles/10.1186/1756-0381-4-10
- Controllability analysis of the directed human protein interaction network identifies disease genes and drug targets. (2016, April). Retrieved February 2023, from https://www.pnas.org/doi/10.1073/pnas.1603992113
- Jingyu Hou. (2017). Protein Function Prediction from Functional Connectivity. Retrieved February 2023, from https://www.sciencedirect.com/topics/mathematics/protein-interaction-network
- • Diana Ekman, Sara Light, Åsa K Björklund,and Arne Elofsson. (2006, June). properties characterize the hub proteins of the protein-protein interaction network of Saccharomyces cerevisiae. Properties characterize the hub proteins of the protein-protein interaction network of Saccharomyces cerevisiae. Retrieved February 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1779539/#:~:text=Protein%20classification%20in%20the%20network&text=Hubs%20are%20defined%20in%20DIP,the%20rest%20are%20inte
- • Yutaka Hata & Junko Iida. (2009). Scaffold Protein. Retrieved February 2023, from https://link.springer.com/referenceworkentry/10.1007/978-3-540-29678-2_5231#author-information

Members Contribution

- Code Implementation
  - Mahmoud Yaser Salman
  - Omar Saad EL-Gharabawy
- Project Research
  - Abdelrahman Ali Farouk
  - Nevein Mohamed Ayman
- Report
  - Abdelrahman Ali Farouk
  - Mahmoud Yaser Salman
  - Nevein Mohamed Ayman
- Slides
  - Omar Saad El-Gharabawy


Members Contribution
Mahmoud
Code Implementation
Slides
Omar
Report
Project Research
Abdelrahman
Nevein