Biological Networks

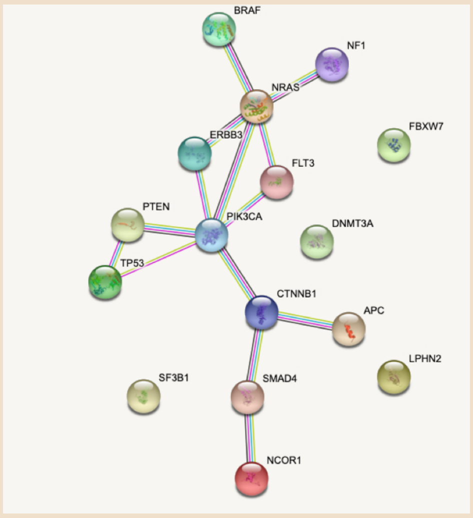
In this class, we will discuss about biological networks and how can they be modelled by using graphs. Different graph frameworks can be used to model different aspects of the network. We will motivate for the use of these data structures in bioinformatics. We will describe the main characteristics that a network can assume and to interpret them. We will implement the required functionality in the class MyGraph.

Learning Objectives

- Motivate for the problem of modelling biological networks with graph structures.
- Applications of biological network analysis.
- Types of Graphs and their representation.
- Graph measures.
- Interpretation of biological network features.

1. Review the slides “Graphs and Biological Networks”.

Task 1 – Characteristics of the Biological Network



This graph was obtained from STRING-DB database and represents the 20 most frequently mutated human cancer genes.

1. Consider a representation of this information as undirect graph and manually calculate the following information:

- Number of nodes and edges in the graph;
- Adjacent nodes of NRAS;
- Node with highest degree;
- Degree distribution for the graph;
- Length of the shortest and longest path between TP53 and BF1;
- DFS and BFS traversal from PIK3CA and NF1 and SMAD4;

Task 2 Complete the remaining methods

2. *get_nodes(self)*: Returns list of nodes in the graph
3. *get_edges(self)*: Returns edges in the graph as a list of tuples (origin, destination)
4. *size(self)*: Returns size of the graph : number of nodes, number of edges
5. *add_vertex(self, v)*: Add a vertex to the graph; tests if vertex exists not adding if it does
6. *add_edge(self, o, d)*: Add edge to the graph; if vertices do not exist, they are added to the graph
7. *out_degree(self, v)*: Number of successors of vertex
8. *in_degree(self, v)*: Number of predecessors of vertex
9. *degree(self, v)*: Unique set of predecessors and successors of vertex
10. *mean_degree(self, deg_type = "inout")*: average degree of all nodes: sum of all degrees divided by number of nodes
11. *prob_degree(self, deg_type = "inout")*: Counting of the number of occurrences of each degree in the network and its frequencies;
12. *print_prob_degree(self, counts)*: Print the degrees and frequencies one per line;

13. *all_clustering_coefs(self)*: Returns the clustering coefficient for all the nodes in the network;
14. *mean_clustering_coef(self)*: Calculates the mean clustering coefficient for the network;

Task 3. Define the graph and call the methods

After completing the code on the class *MyGraph.py (task 2)*, develop a test function that implements the representation of the above graph structure (task 1) and calculates the above information.