Lab activity: analysis of transcriptional regulatory networks

1. Session 2: network analysis

1.1. Introduction

In this second part of the lab activity, we will perform an analysis of the *E. coli* transcriptional regulatory network we generated using RegulonDB data. We will use this analysis to understand how this biological network is organized, what its main properties are, and how these map to the underlying biology. We will also probe into whether and which transcription factors are attractive targets for the development of new *antibiotics*.

1.1.1. Antibiotics

Antibiotics are substances that kill or inhibit the growth of bacterial cells. They do so by targeting an essential component of bacterial cells (e.g. a component of the cell membrane), leading to cell death or stopping cell replication. Contrary to popular belief, humanity has not *developed* any antibiotics; it has merely *discovered* them. Antibiotics are naturally occurring compounds produced predominantly by microorganisms (but also by plants and fungi). The producers of antibiotics are typically the ecological competitors of the bacteria the antibiotics kill, giving them an upper hand in their competition for the resources of that particular ecosystem.

Antibiotic resistance is a rising and important problem. Antibiotic treatments exert a very strong selective pressure on bacteria. This means that any bacteria with a variant of the antibiotic target that is less sensitive to the antibiotic will be rapidly selected for and dominate the population. Bacteria can develop variants of the antibiotic target by random mutation. This is facilitated by the fact that bacterial populations are huge (10 to 30 billion cells per milliliter). Bacteria carrying these mutations will be rapidly selected for and new mutations will make the target even less sensitive, resulting in heavy resistance in a matter of days[1]. On the other hand, bacteria can also acquire mutated versions of the antibiotic target from other bacteria, in a process known as lateral gene transfer (LGT; aka. bacterial sex). Because antibiotics have been produced naturally by microorganisms for millions of years, there is bound to be bacteria resistant to these antibiotics in the wild, facilitating the spread (via LGT) of this resistance once humans start using the antibiotics.

Antibiotic-resistant bacterial infections are projected to kill over 2 million people per year by 2050. This emergent problem makes urgent the development of new antibiotics. Some researchers are working to develop antibiotics that target transcription factors. In this lab activity, you will hopefully be able to determine which transcription factors might be good targets for antibiotics, and why developing this new type of antibiotics might provide advantages over the use of traditional antibiotics.

[1] Check out this beautiful study (https://www.youtube.com/watch?v=plVk4NVIUh8) for a visual depiction of this process.

1.2. Network analysis

In this second session, we will use the graphs generated from the RegulonDB dataset to analyze different properties of transcriptional regulatory networks and obtain some insights on the inherent biology.

1.2.1. Degree distribution

A key parameter that we will analyze is the degree distribution of the *E. coli* transcriptional regulatory network.

1. **(1 point)** Implement a function that, given a network (directed graph), the type of degree to analyze (in-degree, out-degree or general), the type of node we want to analyze (TF, TG or all) and the number of *N* top (highest degree) nodes to be reported, will return a tuple containing: 1) a list with the degree count distribution (counts for every possible degree, from lowest to highest degree) and 2) the dictionary with node names (locus_tags in the *E. coli* network) as keys and their degree values as values, for the top *N* nodes.

```
def degree_distribution(G : nx.DiGraph, degree_type : str, \
                        node_type : str, bestN : int) -> tuple:
    '''Compute the in-, out- or general degree distribution.

    Parameters
    ----------
    - param: G : Networkx graph
        Graph to analyze
    - param: degree_type : str
        Type of degree to compute ('in', 'out', 'general')
    - param: node_type : str
        Type of nodes on which we report degrees ('TG', 'TG', 'all')

    - return: list
        list with degree frequency distribution
    - return: dict
        dictionary with node names as keys and their degrees as values
        for the top N nodes in the degree distribution
    '''
    # ------- IMPLEMENT HERE THE BODY OF THE FUNCTION ------- #
    pass
    # ----------------- END OF FUNCTION --------------------- #
```

1.2.2. Connected components

The inferred *E. coli* transcriptional regulatory network is a directed, disconnected graph. Some of the analyses we want to perform require that the graph be connected, so we will implement a function to obtain the largest connected component of the underlying graph.

2. **(1 point)** Implement a function that, given a network (undirected graph), will return a new network corresponding to the largest connected component of the input graph.

```
def largest_CC_graph(G : nx.Graph) -> nx.Graph:
    '''Generate the largest connected component graph.

    Parameters
    ----------
    - param: G : Networkx graph
```

```
        Graph to analyze
    - return: Networkx graph
        the graph corresponding to the largest connected component in G
    '''
    # ------- IMPLEMENT HERE THE BODY OF THE FUNCTION ------- #
    pass
    # ----------------- END OF FUNCTION --------------------- #
```

1.2.3. Perturbation analysis

Biological systems operate in a noisy environment, challenged by mutations and interactions with the environment and other living entities. One parameter that we are interested in is the resilience to perturbation of the inferred transcriptional regulatory network and, in particular, how the removal of nodes can impact the network. To study this, we will analyze the performance of the network, when faced with node removal, in terms of the average distance among nodes in the network.

To this end, we will serially remove every TF node from the network and analyze how the removal affects the average distance on the network. `NetworkX` provides an `average_shortest_path_length` function, but this function iterates over all the node-pairs in the graph, leading to very long run-times if we perform a serial deletion analysis. Hence, we will implement a function to randomly sample the average shortest distance in the graph.

3. **(1 point)** Implement a function that, given a network (undirected graph) and a number of iterations, will return the average distance in the input graph. At each iteration the function will select two random nodes and compute their distance, returning the average of these computations. Note that node deletion may disconnect parts of the network, leading to infinite distances. If the two selected nodes are not connected, we will ignore them and sample a new pair.

```
def average_distance(G : nx.Graph, iterations : int) -> float:
    '''Estimate the average distance in the input graph.

    Parameters
    ----------
    - param: G : Networkx graph
        Graph to analyze
    - param: iterations : int
        Number of iterations to perform
    - return: float
        the estimated average distance in G
    '''
    # ------- IMPLEMENT HERE THE BODY OF THE FUNCTION ------- #
    pass
    # ----------------- END OF FUNCTION --------------------- #
```

4. **(1 point)** Implement a function that, given a network (undirected graph), a list of nodes, a grouping size and an iteration number, will first compute the network average distance (exact), then serially delete each of possible node groupings of the given size among the listed nodes, estimate the new average distance, compare the resulting network's average distance to the reference average distance, and return a dictionary with tuples of grouping node names as keys and the difference between post-deletion and initial average distances as values.

```python
def deletion_impact(G : nx.Graph, node_list : list,\
                      grouping_size : int, iterations : int) -> dict:
    '''Assess the impact of node deletions on the graph average distance.

    Parameters
    ----------
    - param: G : Networkx graph
        Graph to analyze
    - param: node_list : list
        List of nodes to delete from the network
    - param: grouping_size : list
        The size of the groupings among nodes in the list
    - param: iterations : int
        Number of iterations to perform for average distance
    - return: dict
        Dictionary with grouping node names tuples as keys and differential
        average distance as values.
    '''
    # ------- IMPLEMENT HERE THE BODY OF THE FUNCTION ------- #
    pass
    # ----------------- END OF FUNCTION --------------------- #
```

1.3. Questions to answer in the report

1. **(0.25 points)** Analyze the in-degree distribution of the full *E. coli* (no operons) network. Plot the distribution on a linear and log-log scale and attach the plots. What statistical distribution do you believe approximates best the observed degree distribution (you can perform statistical fitting if needed)? What is the mean value of the observed distribution, and what does this tell us about the input-output mapping of regulation for most genes in *E. coli*?

2. **(0.25 points)** Analyze now the out-degree distribution of the *E. coli* (no operons) network. Plot the distribution on a linear and log-log scale and attach the plots. What statistical distribution do you believe approximates best the observed degree distribution. Based on these results, what kind of network could we say the *E. coli* transcriptional regulatory network is?

3. **(0.5 points)** Analyze now the in-degree distribution of the entire *E. coli* (with operons) network. Compute and plot the difference in degree distribution (counts in network with operons minus counts in networks without operons). Is there a discernible peak in the difference distribution? What could this be telling us about operon organization in *E. coli*?, thinking of an operon as a programming "function"?

4. **(0.1 points)** Let's now analyze the top 10 nodes in the in-degree distribution (highest in-degree). List their locus_tag, gene names and in-degree.

5. **(0.3 points)** Search EcoCyc or RegulonDB with the locus_tags of these 10 genes (or Google up their names) to gain some information on the function of these genes. You should find the name of a particular metabolite (an amino acid) appearing several times

in the summary description of these genes. A metabolite is a substance that participates in metabolism, the set of chemical reactions in cells that change food into energy (catabolism) or into building blocks to create complex molecules (anabolism). Look up this metabolite to learn about its importance in bacteria. Does it make sense that several high in-degree nodes are linked to it?

6. **(0.1 points)** How many of these are described as regulators?

7. **(0.3 points)** What biological processes do the two regulators with the highest in-degree regulate (you can look at their Gene Ontology (GO) Terms [Biological Process] for clues; ignore all processes related only to transcriptional regulation)? Look up concepts associated with these processes. Does it make sense that these regulators have such a high in-degree?

8. **(0.2 points)** Report on the 10 transcription factors with the highest out-degree. List their locus_tag, gene names and out-degree. Look up the function for the three TF with the highest out-degree. What general process of the cell are these linked to?

We are interested in finding out if the *E. coli* transcriptional regulatory network has small-world properties. This requires analyzing the clustering coefficient and average distance in the network. Unfortunately, the *E. coli* transcriptional regulatory network is a disconnected directed graph, so we will focus on the largest connected component of its underlying graph to perform the analysis.

9. **(0.1 points)** Obtain the largest connected component of the underlying graph corresponding to the *E. coli* (no operons) network. Report on its order and size. Do you consider this network to be representative of the overall network?

10. **(0.1 points)** Compute and report on the average clustering coefficient, average distance and diameter on the largest connected component of the underlying graph corresponding to the *E. coli* (no operons) network.

11. **(0.6 points)** Test whether these values are larger than we would expect by comparing them to those of randomly generated graph. Document the models and parameters that you try and why you try them, and the values obtained. Does the *E. coli* transcriptional regulatory network appear to be optimized for these metrics?

12. **(0.7 points)** Do you believe these properties stem from the specific distribution of degrees in the *E. coli* transcriptional regulatory network? How could you test this

hypothesis? Report on any experiments that you perform on this issue and your interpretation of them.

13. **(0.2 points)** Test your `average_distance` function on the largest connected component of the underlying graph corresponding to the *E. coli* (no operons) network. Compare it to the reference `average_shortest_path_length` function results. How many iterations are required to get a reasonable approximation (i.e. two decimal places)? Report on the trials performed. What is the *theoretical* speed up (in terms of number of shortest path computations) from using this function with the number of iterations required to obtain a good approximation?

14. **(0.3 points)** Perform a perturbation analysis of the largest connected component of the underlying graph corresponding to the *E. coli* (no operons) network, using individual nodes, for the set of transcription factor (TF) nodes that belong to this component. Report on the 10 transcription factors with most impact on average distance. List their locus_tag, gene names and average distance. Contextualize these results based on your previous analysis of degree distribution.

15. **(0.4 points)** Think of a network metric that you believe should correlate well with the perturbation analysis. Justify your answer and report on whether this metric correlates well with the perturbation analysis for transcription factors.

16. **(0.4 points)** Perform now a perturbation analysis based on pairs of nodes for the set of 30 transcription factor (TF) nodes with highest impact. Report on the results obtained (5 most impactful pairs). Are they additive with respect to single node perturbation? What does this tell us about the *E. coli* transcriptional regulatory network?

17. **(0.3 points)** Based on our analysis, if you had to choose a transcription factor to partially disable a cell, which one would you choose? In terms of antibiotic resistance, what could be an advantage of using antibiotics that target transcription factors?

18. **(0.6 points)** Optimizing a network for specific properties necessarily leads to a tradeoff. For instance, low average distance typically relies on the presence of hubs, but this might render the network vulnerable to targeted attacks. Perform any additional comparisons you can think of with random models to assess whether this is true for the largest connected component of the underlying graph corresponding to the *E. coli* (no operons) network. Report on any comparisons made and the interpretation of the results obtained.

19. **(0.3 points)** So far, no naturally occurring antibiotics that target transcription factors have been discovered. Consider why, and whether this might be a good thing if people start developing them.