NetworkMetrics.py

Proximity, Separation and LCC Metrics in Graphs.

This module provides a collection of functions to compute proximity, separation and Largest Connected Component (LCC) metrics between sets of nodes in a graph. The metrics have been developed based on various research works, including the approach by Menche et al., 2015.

The main functions in this module include: - `proximity`: Computes the proximity between two sets of nodes in a graph. - `separation`: Calculates the separation between two sets of nodes in a network. - `separation_z_score`: Determines the z-score and p-value of the separation between two node sets based on randomized samples. - `lcc_significance`: Determines the z-score and p-value of the size of an LCC based on randomized samples.

These functions use both exact and approximate methods for degree-preserving randomization of node sets. Additionally, precomputed distance matrices can be leveraged for efficient computation.

Authors:

- Andres Aldana Gonzalez (a.aldana@northeastern.edu)
- Rodrigo Dorantes Gilardi (r.dorantesgilardi@northeastern.edu)

Creation Date: October 4, 2023 Last Modification: October 18, 2023

References:

- Menche, Jörg, et al. “Uncovering disease-disease relationships through the incomplete interactome.” Science 347.6224 (2015). DOI 10.1126/science.1257601
- Guney, Emre, et al. “Network-based in silico drug efficacy screening.” Nature Communications 7,1 (2015). DOI 10.1038/ncomms10331

Functions

get_avg_min_shortest_path_dmatrix(net,A,B,D):

Returns the average minimum distance between each node in A and all nodes in B, using the distance matrix D to access precomputed distances between nodes.

Parameters

net : networkx.Graph The input network/graph for which distances need to be computed.

A : Iterable (list, set, etc.) A collection of nodes from which the shortest paths to nodes in B will be computed.

B : Iterable (list, set, etc.) A collection of nodes to which the shortest paths from nodes in A will be computed.

D : dict of dicts A distance matrix where D[i][j] gives the precomputed shortest distance between nodes i and j. If there’s no path between i and j, D[i][j] should be None.

Returns

avg : float The average of the minimum distances between each node in A and all nodes in B.

all_pair_distances(graph):

Calculates the shortest path distance between every pair of nodes in the graph and returns a dictionary where keys are nodes and values are dictionaries of distances to other nodes. NOTE: This function utilizes multiple processes. Ensure it's invoked from the main execution environment only.

Parameters

graph : networkx.Graph The input graph for which pairwise distances will be computed. Nodes should be unique and hashable.

Returns

dict: A nested dictionary where the outer keys are nodes of the graph and the inner keys are nodes reachable from the outer node. The value at `dict[node1][node2]` gives the shortest path distance from `node1` to `node2`. If `node2` is not reachable from `node1`, it may not be present as an inner key for `node1`.

save_distances(distances, filename):

Saves the precomputed distance matrix to a file using the `pickle` module.

This function serializes the given distance matrix and writes it to a specified file. The saved file can later be loaded to quickly retrieve the distance matrix without needing to recompute the distances.

Parameters:

distances : dict of dicts The distance matrix represented as a nested dictionary. The outer keys are source nodes, the inner keys are target nodes, and the values are the shortest path distances from the source to the target. E.g., distances[node1][node2] gives the shortest path distance from `node1` to `node2`.

filename : str The path and name of the file to which the distances should be saved. If the file already exists, it will be overwritten.

Notes:

The saved file is in binary format due to the usage of the `pickle` module. Always be cautious when loading pickled data from untrusted sources as it can be a security risk.

load_distances(filename):

Loads a precomputed distance matrix from a file using the `pickle` module.

This function deserializes and retrieves a distance matrix (in the form of a nested dictionary) that was previously saved to a file. It's the inverse operation to saving the matrix using the `pickle` module.

Parameters:

filename : str The path and name of the file from which the distance matrix should be loaded. The file should have been previously saved using the `pickle` module, via the `save_distances` function.

Returns:

dict of dicts: The distance matrix represented as a nested dictionary. The outer keys are source nodes, the inner keys are target nodes, and the values are the shortest path distances from the source to the target. For instance, distances[node1][node2] gives the shortest path distance from `node1` to `node2`.

Notes:

The loaded file is in binary format due to the usage of the `pickle` module. Always exercise caution when loading pickled data from untrusted sources as it can present a security risk.

get_binning(net, bin_size=100):

Return a histogram of the degrees of the PPI.

The histogram should have bins with size at least equal to `bin_size`. For each bin, the bin bounds l and u should be optimized such that a bin with bounds l and u - 1 does is of size smaller than `bin_size`. Original code: Rodrigo Dorantes

Note that for a node with degree d to be in a bin with bounds (l, u], we have that l < d <= u.

Parameters

net: proximity.Network Usually the protein-protein interaction network. **NOTE** If `net` is a gt.Graph instance, it should have a `gt.VertexPropertyMap` with the node names caled "ids". bin_size: int

Returns

nodes: list. The nodes of each bin.

lower: list. The lower bound of each bin

upper: list. The upper bound of each bin.

proximity(net,T,S,D,degree_preserving = 'exact',n_iter=1000,bin_size=100,symetric=False):

Calculates the proximity between two sets of nodes in a given graph.

The function defines the proximity between two sets of nodes based on the paper by Guney et al. 2016. Link: doi:10.1038/ncomms10331

Proximity uses either the average shortest path length (ASPL) between two sets or the symmetrical version (SASPL), which averages the distances in both directions (i.e., ASPL(A, B) and ASPL(B, A)).

Parameters

net : networkx.Graph The input graph for which pairwise proximities will be computed.

T : Iterable (list, set, etc.) A collection of 'source' nodes.

S : Iterable (list, set, etc.) A collection of ‘target’ nodes.

D : dict of dicts A precomputed distance matrix where D[i][j] provides the shortest distance between nodes i and j.

degree_preserving : str, optional Method for degree-preserving randomization. Valid values are ‘exact’ and ‘log-binning’. ‘exact’ uses accurate degree-preserving randomization, while ‘log-binning’ employs an approximation based on a logarithmic binning of node degrees. Default is ‘exact’.

n_iter : int, optional Number of iterations/samples for assessing significance. Default is 1000.

bin_size : int, optional Determines the size of the logarithmic bins when using the ‘log-binning’ method. Default is 100.

symetric : bool, optional If set to True, the function computes the symmetrical version of proximity, using SASPL. Otherwise, it uses ASPL. Default is False.

Returns

dict: A dictionary containing various statistics related to proximity, including:

- ‘d_mu’: The average distance in the randomized samples.
- ‘d_sigma’: The standard deviation of distances in the randomized samples.
- ‘z_score’: The z-score of the actual distance in relation to the randomized samples.
- ‘p_val’: The p-value corresponding to the z-score.
- ‘raw_amspl’: The raw average minimum shortest path length between sets T and S.
- ‘dist’: A list containing distances from each randomization iteration.

Notes

Ensure the network is connected for meaningful proximity values. Disconnected components may skew results.

separation(net,A,B,D):

Computes the separation between two sets of nodes A and B in the network `net` as defined by Menche et al., 2015.

The separation is a measure that indicates how distantly or closely two sets of nodes (e.g., genes or proteins) are located in a network. It can be useful for understanding disease-disease relationships or interactions between different groups of nodes in a biological network.

Parameters

net : networkx.Graph The input network or graph in which the separation between node sets A and B will be calculated.

A : container (list, set, etc.) A subset of nodes in `net` representing the first group.

B : container (list, set, etc.) A subset of nodes in `net` representing the second group.

D : dict of dicts A precomputed distance matrix where D[i][j] provides the shortest distance between nodes i and j. This matrix should be generated using the `all_pair_distances` function or an equivalent method.

Returns

float. The separation value between node sets A and B in the network `net`. A smaller value indicates that the two sets are closer in the network, while a larger value indicates that they are more distantly located.

References

[1] Menche, Jörg, et al. “Uncovering disease-disease relationships through the incomplete interactome.” Science 347.6224 (2015).

separation_z_score(net,A,B,D,degree_preserving=‘exact’, n_iter=‘1000’,bin_size=‘100’):

Calculates the z-score of the separation between two sets of nodes A and B in the network `net` based on randomized node sets with degree-preserving properties.

This function first calculates the actual separation between node sets A and B. It then derives a reference distribution of separations by performing degree-preserving randomizations of the node sets. The resulting z-score gives how many standard deviations the actual separation is from the mean of the reference distribution.

Parameters

net : networkx.Graph The input network or graph in which the separation between node sets A and B will be assessed.

A : container (list, set, etc.) A subset of nodes in `net` representing the first group.

B : container (list, set, etc.) A subset of nodes in `net` representing the second group.

D : dict of dicts A precomputed distance matrix where D[i][j] gives the shortest distance between nodes i and j.

degree_preserving : str, optional Method for degree-preserving randomization. Options are ‘exact’ and ‘log_binning’. ‘exact’ ensures precise degree preservation, while ‘log_binning’ employs an approximation based on logarithmic binning of node degrees. Default is ‘exact’.

n_iter : int, optional Number of random sampling iterations used to derive the reference distribution of separations. Default is 1000.

bin_size : int, optional Determines the size of the logarithmic bins when using ‘log_binning’. Relevant only if `degree_preserving` is set to ‘log_binning’. Default is 100.

Returns

dict

A dictionary containing:

- ‘d_mu’: Mean separation from the randomized samples.
- ‘d_sigma’: Standard deviation of separations from the randomized samples.
- ‘z_score’: Z-score of the actual separation against the randomized samples.
- ‘p_val’: The p-value corresponding to the z-score.
- ‘raw_separation’: Actual separation value between node sets A and B.
- ‘dist’: List of separations from each randomization iteration.

Notes

The degree-preserving randomization ensures that the randomized node samples have a degree distribution similar to the original sets, ensuring a fair comparison.

lcc_significance(net, A, degree_preserving='exact', n_iter='1000',bin_size='100')

Calculate the statistical significance of the size of the Largest Connected Component (LCC) of a subgraph induced by node set `A` in the network `net`.

This function generates a null model distribution for the LCC size by resampling nodes from the network while preserving their degrees. The statistical significance of the observed LCC size is then determined by comparing it against this null model distribution.

Parameters:

net : networkx.Graph The input network.

A : list or set The set of nodes for which the LCC is to be determined.

degree_preserving : str, optional (default='exact') The method used to preserve node degrees during resampling. Can be 'exact' or 'log_binning'.

n_iter : int, optional (default=1000) Number of iterations for generating the null model distribution.

bin_size : int, optional (default=100) Size of bins if 'log_binning' method is used.

Returns:

dict :

A dictionary containing:

- 'd_mu': Mean of the null model LCC size distribution.
- 'd_sigma': Standard deviation of the null model LCC size distribution.
- 'z_score': The z-score of the observed LCC size.
- 'p_val': The p-value corresponding to the z-score.
- 'lcc': Nodes in the largest connected component of `A`.
- 'lcc_size': Size of the largest connected component of `A`.
- 'dist': The null model LCC size distribution.

Raises:

Exception: If the 'degree_preserving' method is neither 'exact' nor 'log_binning'.

Notes:

- Make sure the network does not contain any isolated nodes.