—

Welcome to coexpressiongraph’s documentation!

Module for Coexpression Network Extraction

This module takes as input several gene profiles and calculates an undirected graph based on the coexpression between them, calculated using one of the following metrics:

- Pearson correlation coefficient (‘PCC’)
- Biweight Midcorrelation (‘BICOR’)
- Distance correlation (‘DCORR’)

General algorithm

The general idea behind the module can be summarized as follows:

1. Read expression file and apply log_10 if required
2. Calculate coexpression metric (in parallel using multiprocessing)
3. Calculate bins for quantification
4. Generating auxiliary plots (optional)
5. Finding threshold based on density and marginal addition of nodes
6. Apply threshold to find coexpression graph

Reading the input file

The input file for *coexpression_graph* is loaded using *pandas*, which infers automatically if the file has headers. However, there is a possibility to specify the file delimiter using the parameter **sep**, which by default is *None*.

It is also optional to apply a logarithm (base 10) to the expression values before calculating the correlation metric. The corresponding parameter for this effect is **lognormalize_input**. By default, this option is set to *False*.

Calculating the Threshold

The selection of the threshold is calculated as follows:

1. The selected metric is calculated for all posible pairs of gene profiles.
2. A number of bins is created and the network properties are quantified using those bins. The size of each bin is 0.01, ranging from the minimum to the maximum values calculated previously.
3. Two bins are of importance: the one where the minimum of network density occurs, and the one which has a higher increase in the number of nodes included in the network.
4. The minimum of these two bin values is selected as the threshold.
5. Those pairs whose metric value is greater than or equal to the threshold belong to the coexpression graph.

The idea behind the threshold selection is based on the following papers:

- van Verk, M. C., Bol, J. F., & Linthorst, H. J. (2011). Prospecting for genes involved in transcriptional regulation of plant defenses, a bioinformatics approach. *BMC plant biology* , 11, 88. https://doi.org/10.1186/1471-2229-11-88

- Zhang, L., Yu, S., Zuo, K., Luo, L., & Tang, K. (2012). Identification of gene modules associated with drought response in rice by network-based analysis. *PloS one* , 7(5), e33748. https://doi.org/10.1371/journal.pone.0033748

Saving auxiliary files

It is possible to save some files which can help in visualizing properties such as density, number of nodes, number of edges, etc. The parameter **save_files** is set by default to *None*, but can be set to a string indicating the path where the files should be stored.

The stored files are:

- **Plots**: The saved plots as a function of the metric value are the size of the greatest connected component (*{metric}-CCmaxsize.png*), the network density (*{metric}-Density.png*), the number of edges (*{metric}-Edges.png*) and the number of nodes (*{metric}-Nodes.png*).
- **Metric values**: Files of the form *{metric}-{i}.txt* are generated, including the correlation information.
- **Coexpression graph**: The file *{metric}trimmed.csv* contains the generated network, after the threshold.
- **Bins**: The file *{metric}-bins.csv* can be used to generate the plots again. It includes a header and a line for each bin, specifying the corresponding values.

API Reference

`coexpression_graph.`**`coexpression_graph`**`(`*`file, metric='PCC', save_files=None, lognormalize_input=False, num_threads=4, sep=None`*`)`

Given a file containing n rows of gene expression profiles with m accessions each, it applies the given metric ('PCC', 'BICOR', 'DCORR') to calculate a coexpression matrix. Finally, a histogram is calculated with bins of 0.01 and a threshold is calculated to create a coexpression graph.

The threshold is obtained based on two references:

1. The bin which causes minimum network density
2. The bin which has the highest marginal addition of nodes to the network, that is, whose difference in nodes with the previous bin is maximal.

Parameters:

- **file** (*str*) – Path to the file containing the expression profiles
- **metric** (*str*) – Metric to be applied to correlate expression profiles: 'PCC' for Pearson correlation coefficient, 'BICOR' for Biweighted Midcorrelation, and 'DCORR' for Distance correlation. Default: 'PCC'
- **save_files** (*None or str*) –
  Wether the intermediate files should be stored in disk. If value is _None_, files are not saved in disk. Otherwise, a string indicating the path should be given. The intermediate files which are stored are:
  - Given a expression matrix with n genes, each with m expression values, `n-1` files with the name *{metric}-{i}.txt* are stored containing the upper diagonal coexpression matrix. Each file will have the coexpression values using **metric** for gene *i* with every other *j* genes, for 0 <= i < j < n.
  - File *{metric}-{i}.txt* (e.g. PCC-0.txt) will be a comma-separated file including each *j* and the coexpression value between *i* and *j*.
  - Plots of the form *{metric}-{parameter}.png*, where parameter can be Density, CCmaxsize, Nodes and Edges

  - The trimmed network: *{metric}trimmed.csv*.
- **lognormalize_input** (*bool*) – Whether a logarithm base 10 will be applied to expression values before calculating the coexpression metric
- **num_threads** (*int*) – Number of processes to be used in parallel when calculating coexpression.
- **sep** (*None or str*) – The string used to separate values.

Returns: E: The coexpression matrix as an edge list. Each value in the list will be a tuple with three values: the first two values are the zero-indexed positions of the genes and the third value is the corresponding coexpression value between the gene expression profiles.

Return type: List[tuple]

Module for Affinity Matrix Computation

Create new distance matrix using information from the gene co-expression network and information from the associations between genes and functions. The new distance between two genes will be the mean of the weight between two nodes and the proportion of shared functions. Please load the coexpression data without labels. Dataframe between genes and functions gene_by_func must have gene ID in the first column and an array of functions in the second column.

API Reference

`affinity_matrix.`**`affinity_matrix`**`(`*`edgelist, gene_by_func, normalize, save`*`)`

The maximum and minimum values of co-expression are saved and two dictionaries are created to optimize searches

Parameters:

- **edgelist** (*DataFrame*) – Coexpression matrix as an edge list. Each value in the list will be a tuple with three values: the first two values are the genes associated and the third value is the corresponding coexpression value between them. *source* and *target* are of type *int*, while *score* is *float*
- **gene_by_func** (*DataFrame*) – The matrix with the data of functions associated with a gene, this matrix must have gene ID in the first column and an array of all its functional annotations in the second column identified with the GO term.
- **normalize** (*bool*) – The coexpression values given by the edge list are normalized.
- **save** (*bool*) – The affinity matrix is saved as a csv archive, else return the new variable.

Returns: The affinity matrix as a new relationship value between genes is returned.

Return type: Matrix

`affinity_matrix.`**`test`**`()`

Charge the matrix with the data of functions associated with a gene, this matrix must have gene ID in the first column and an array of all it's functions in the second column

Indices and tables

- Index
- Module Index
- Search Page