BiMat Use case using Moebus cross-infection matrix data

Table of Contents

This use case will introduce the user to the most basic functionalities of the BiMat Software. In order to do that we will calculate some of the results presented on the Flores et al 2012 paper (Multi-scale structure and geographic drivers of cross-infection within marine bacteria and phages). We will show how to perform the next things:

Add the source to the matlab path

```
g = genpath('matlab'); addpath(g);
close all;
```

We need also to load the data from which we will be working on

```
load moebus_use_case.mat;
```

Creating the Bipartite network object

```
bp = Bipartite(moebus.weight_matrix);    % Create the main object
bp.row_labels = moebus.bacteria_labels;  % Updating node labels
bp.col_labels = moebus.phage_labels;
bp.row_ids = moebus.bacteria_stations;   % Updating node ids
bp.col_ids = moebus.phage_stations;
```

We can print the general properties of the network with:

```
bp.printer.PrintGeneralProperties();
```

```
        General Properties
          Number of species:           501
          Number of row species:       286
          Number of column species:    215
          Number of Interactions:     1332
          Size:                      61490
          Connectance or fill:       0.022
```

Calculating Modularity

The modularity algorithm is encoded in the property modules of the bipartite object (`bp.modules`). Tree algorithms are available:

1. Adaptive BRIM (`AdaptiveBrim.m`)
2. LP&BRIM (`LPBrim.m`)
3. Newman Algorithm (`NewmanAlgorithm.m`)

Adaptive BRIM algorithm is assigned by default during the creation of the Bipartite object. However, we can assign another algorithm dinamically. For example to change to Newman Algorithm:

```
bp.modules = NewmanModularity(bp.adjacency);
bp.modules.DoKernighanLinTunning = true; % This flag is exclusive of the Newman Al
bp.modules = AdaptiveBrim(bp.adjacency); % Return to the default algorithm
```

We need to calculate the modularity by calling:

```
bp.modules.Detect();
```

If we are interested only in node community indexes we can use `bp.modules.row_modules` and `bp.modules.col_modules`. However for modularity values we need to call `bp.modules.Qb` or `bp.modules.Qr` as is: the next lines of code:

```
fprintf('The modularity value Qb is %f\n', bp.modules.Qb);
fprintf('The fraction of interactions inside modules Qr is %f\n',bp.modules.Qr);
```

```
    The modularity value Qb is 0.786410
    The fraction of interactions inside modules Qr is 0.900150
```

The value 0<=Qb<=1 is calculated using the standard bipartite modularity function (introduced by Barber):

$$Q_b = \frac{1}{m} Trace(R^T(B - P)T)$$

while the value 0<=Qr<=1 represents the fraction of interactions that fall inside modules:

$$Q_r = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} B_{ij}\delta(g_i, g_j)$$

Calculating Nestedness

Two algorithms exist for calculating nestedness. Contrary to the case of modularity, in this case there is no need to change the algorithm because all the algorithms have an independent property in the Bipartite object. These algorithms are:

- NODF (Nestedness metric based on overlap and decreasing filling). With value in the interval [0,1].
- NTC (Nestedness Temperature Calculator) With value in the interval [0 1].

The first algorithm is runned during the creation of the Bipartite object, but because the NTC algorithm is slow, you need to run the algorithm explicitally:

```
bp.ntc.CalculateNestedness();
```

Finally to show acces the values of the two algorithms you need to call:

```
fprintf('The nestedness NODF value is %f\n', bp.nodf.nodf); %The same value will b
fprintf('The nestedness NTC value is %f\n', bp.ntc.N); %Because the value depends
```

We can print all the values of structure values by just calling:

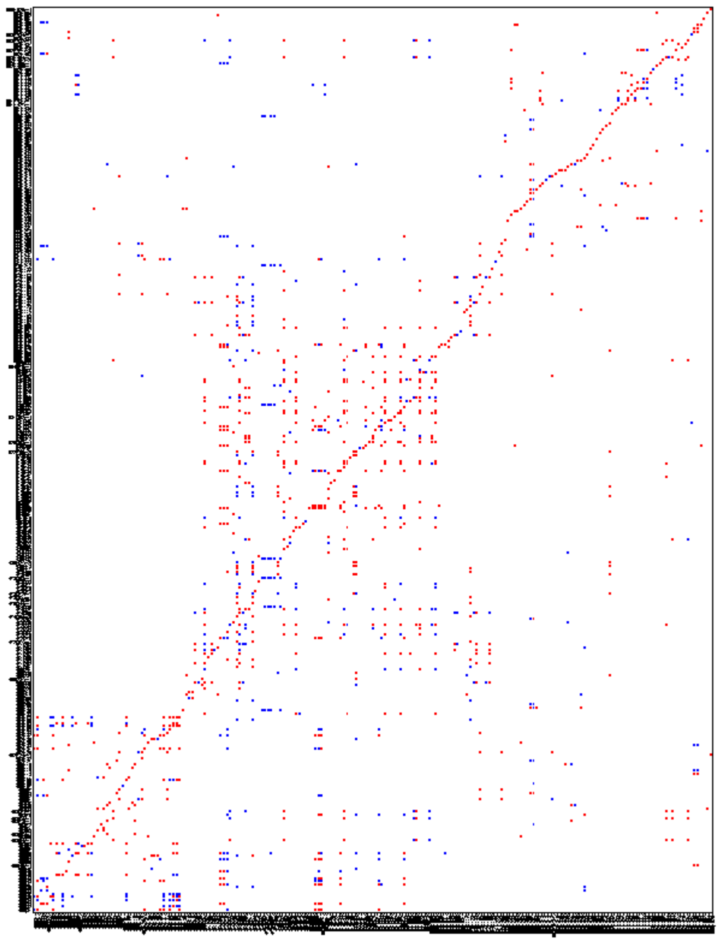
```
bp.printer.PrintStructureValues
```

```
    The nestedness NODF value is 0.034053
    The nestedness NTC value is 0.954006
    Modularity
      Used algorithm:                    AdaptiveBrim
      N (Number of modules):                       47
      Qb (Standard metric):                    0.7864
      Qr (Ratio of int/ext inter):             0.9002
    Nestedness
     NODF metric value:                        0.0341
     NTC metric value:                         0.9539
```

Plotting in Matrix Layout

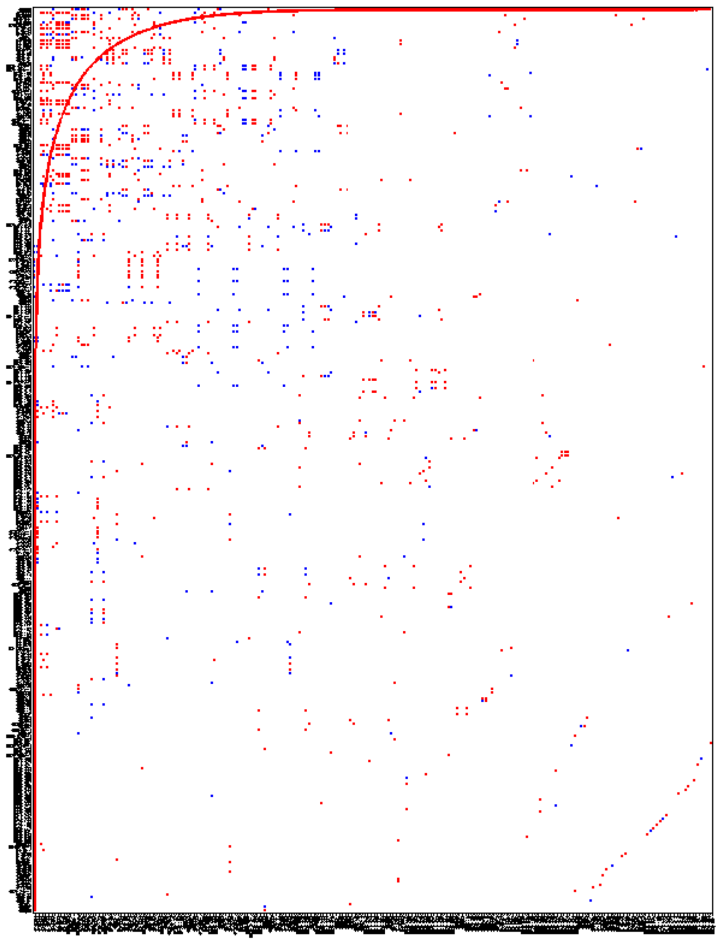
You can print the layout of the original, nestedness and modular sorting. If you matrix is weighted in a categorical way using integers (0,1,2...) you can visualize a different color for each interaction, where 0 is no interaction. For using this functionality you need to assign a color for each interaction and specifically indicate that you want a color for each interaction before calling the plot function:

```
figure(1);
set(gcf,'Position',[19    72   932   922]); % Matlab command to change the figure
bp.plotter.FontSize = 2.5; %Change the font size of the rows and labels
bp.plotter.use_type_interaction = true; % I want to use different color for each k
bp.plotter.color_interactions(1,:) = [1 0 0]; %Red color for clear lysis
bp.plotter.color_interactions(2,:) = [0 0 1]; %Blue color for turbid spots
bp.plotter.back_color = 'white';
bp.plotter.PlotMatrix(); %After changing all the format we finally can call the pl
```



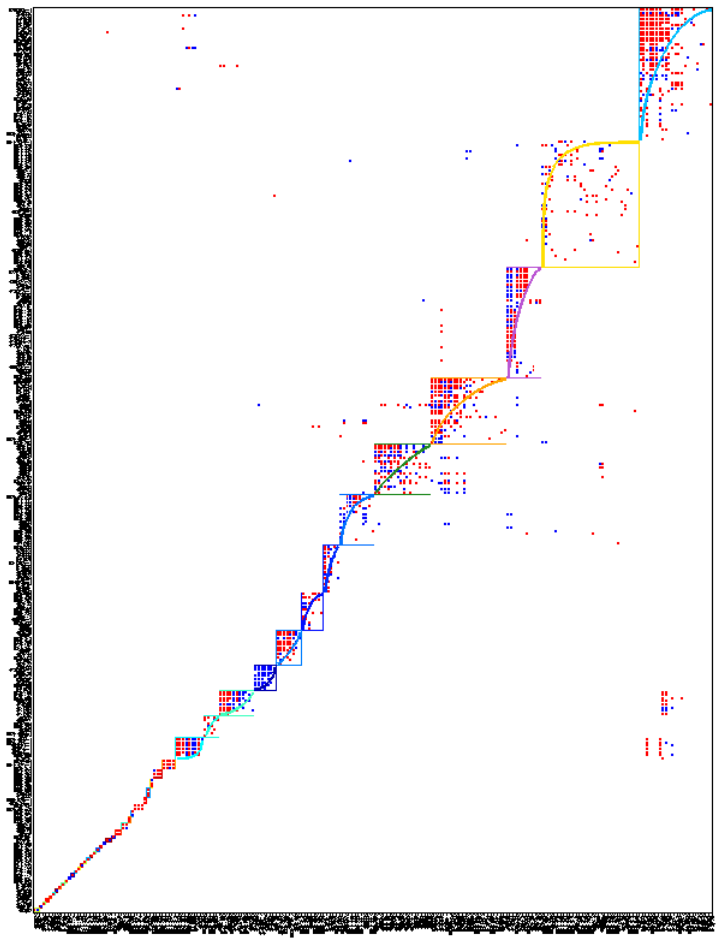
For plotting the nestedness matrix you may decide to use or not an iscoline. The nestedness pattern is just the matrix sorted in decreasing degree for row and column nodes.

```
figure(2);
set(gcf,'Position',[19+200    72   932   922]); % Matlab command to change the fig
bp.plotter.use_isocline = true; % The isocline is used in the NTC algorithm.
bp.plotter.isocline_color = 'red'; %Decide the color of the isocline.
bp.plotter.PlotNestedMatrix();
```



For plotting the modularity sort, the plot function will calculate the modularity (call `bp.modules.Detect()`) if you have not previouslly call it.

```
figure(3);
set(gcf,'Position',[19+400    72   932   922]); % Matlab command to change the fig
bp.plotter.plot_iso_modules = true; %This will plot isoclines inside modules.
bp.plotter.PlotModularMatrix();
```



Statistical analysis in the entire network

We can perform an statistical analysis in the entire network for nestedness and modularity. In order to make an statistical analysis of the structure values we need to decide how many replicates we will need and what null model is more convenient for what we need. File `NullModels.m` contain all the availaible null models, while file `Test.m` contains all the functions required for performing this analysis. We can perfom the analysis in all the structure values by the use of a single function call the required functions for nestedness and modularity analysis:

```
bp.tests.print_output = 0; % Do not show outputs
bp.tests.DoCompleteAnalysis(10, @NullModels.EQUIPROBABLE); %Bernoulli
%bp.tests.DoCompleteAnalysis(10, @NullModels.AVERAGE); %Probability degree
bp.printer.PrintStructureStatistics(); %Print the statistical values
```

```
    Modularity
      Used algorithm:                          AdaptiveBrim
      Null model:                  NullModels.EQUIPROBABLE
      Replicates:                                        10
      Qb value:                             0.7864
      z-score:                             62.9766
      percent:                            100.0000
    NODF Nestedness
      Null model:           NullModels.EQUIPROBABLE
      Replicates:                               10
      NODF value:                           0.0341
      z-score:                             15.6156
      percent:                            100.0000
    NTC Nestedness
      Null model:           NullModels.EQUIPROBABLE
      Replicates:                               10
      NODF value:                           0.9538
      z-score:                             14.9372
      percent:                            100.0000
```

All structure statistics calculate the next numbers:

1. value: Structure value of the tested bipartite network
2. p: p-value of a t-test
3. ci: Confidence interval of the mean of the random bipartite networks using a t-test
4. percent: Ratio of how many random matrices have an structure value smallen than the tested network
5. z-score: Z-score of the value in the tested network using the values of the random networks. Be careful when using this value because the random distribution may not be normal.
6. replicates: The quantity of random networks used in the analysis.

Statistical Analysis of the internal modules

In addition to be able to perform structure analysis in the entire network, we may be able (depending in the size and module configuration of the tested matrix) to perform structure analysis in the internal modules. We will show next (i) how to do an analysis of modularity and nestedness in the internal modules and (ii) how to test for a possible correlation between node labeling and module configuration. All the functions for performing this kind of analysis is encoded in file `TestModules.m`. For calculating the the statistical structure of the internal modules we just need to call:

```
bp.testmodule.TestInternalModules(20,@NullModels.EQUIPROBABLE); %100 replicates an
bp.printer.PrintStructureStatisticsOfModules(); % Print the results
```

```
  Testing Matrix: 1 . . .
  Testing Matrix: 2 . . .
  Testing Matrix: 3 . . .
  Testing Matrix: 4 . . .
  Testing Matrix: 5 . . .
  Testing Matrix: 6 . . .
  Testing Matrix: 7 . . .
  Testing Matrix: 8 . . .
  Testing Matrix: 9 . . .
  Testing Matrix: 10 . . .
  Testing Matrix: 11 . . .
  Testing Matrix: 12 . . .
```

```
Testing Matrix: 13 . . .
Testing Matrix: 14 . . .
Testing Matrix: 15 . . .
Testing Matrix: 16 . . .
Testing Matrix: 17 . . .
Testing Matrix: 18 . . .
Testing Matrix: 19 . . .
Testing Matrix: 20 . . .
Testing Matrix: 21 . . .
Testing Matrix: 22 . . .
Testing Matrix: 23 . . .
Testing Matrix: 24 . . .
Testing Matrix: 25 . . .
Testing Matrix: 26 . . .
Testing Matrix: 27 . . .
Testing Matrix: 28 . . .
Testing Matrix: 29 . . .
Testing Matrix: 30 . . .
Testing Matrix: 31 . . .
Testing Matrix: 32 . . .
Testing Matrix: 33 . . .
Testing Matrix: 34 . . .
Testing Matrix: 35 . . .
Testing Matrix: 36 . . .
Testing Matrix: 37 . . .
Testing Matrix: 38 . . .
Testing Matrix: 39 . . .
Testing Matrix: 40 . . .
Testing Matrix: 41 . . .
Testing Matrix: 42 . . .
Testing Matrix: 43 . . .
Testing Matrix: 44 . . .
Testing Matrix: 45 . . .
Testing Matrix: 46 . . .
Testing Matrix: 47 . . .
Network,      Qb,Qb mean,Qb z-score,Qb percent,   NODF,NODF mean,NODF z-sc
      1, 0.29035,0.26237,    4.0661,       100,0.53237,  0.29636,     19.2
      2, 0.74015,0.50666,   14.4469,       100,0.10842,  0.10087,     0.65
      3, 0.38129,0.28746,    7.0028,       100,0.36731,  0.34374,      1.4
      4, 0.24574,0.25725,   -1.3976,        10,0.57939,  0.35514,      9.2
      5, 0.29705,0.22951,    7.3274,       100,0.51288,  0.43935,      2.5
      6, 0.51229,0.36437,    6.1788,       100,0.26905,  0.29901,    -0.72
      7, 0.37778,0.32294,    1.9288,       100,0.38468,  0.36374,     0.31
      8, 0.46091,0.36509,    2.3994,        95,0.32567,  0.32563,  0.00055
      9, 0.16272,0.19896,   -3.0853,         0,0.76951,  0.57759,      3.4
     10, 0.12558,0.16321,   -4.1318,         0, 0.6966,  0.63718,     0.80
     11, 0.15174,0.17105,   -2.1231,         5,0.74994,  0.61393,      2.9
     12,     0.4,0.30644,    2.4462,       100,0.41398,  0.42454,   -0.097
     13,0.066639,0.12164,   -7.4192,         0,0.38596,  0.66623,     -3.8
     14,       0,       0,       NaN,         0,      0,        0,
     15, 0.16327,0.16327,  -0.97468,         0,0.66667,  0.66667,     0.97
     16,       0,       0,       NaN,         0,      0,        0,
     17,       0,       0,       NaN,         0,      0,        0,
     18,       0,       0,       NaN,         0,      0,        0,
          19,         0,        0,         NaN,          0,        0,          0,
          20,         0,        0,         NaN,          0,        0,          0,
          21,         0,        0,         NaN,          0,        0,          0,
          22,         0,        0,         NaN,          0,        0,          0,
          23,         0,        0,         NaN,          0,        0,          0,
          24,         0,        0,         NaN,          0,        0,          0,
          25,         0,        0,         NaN,          0,        0,          0,
          26,         0,        0,         NaN,          0,        0,          0,
          27,         0,        0,         NaN,          0,        0,          0,
          28,         0,        0,         NaN,          0,        0,          0,
          29,         0,        0,         NaN,          0,        0,          0,
          30,         0,        0,         NaN,          0,        0,          0,
          31,         0,        0,         NaN,          0,        0,          0,
          32,         0,        0,         NaN,          0,        0,          0,
          33,         0,        0,         NaN,          0,        0,          0,
          34,         0,        0,         NaN,          0,        0,          0,
          35,         0,        0,         NaN,          0,        0,          0,
          36,         0,        0,         NaN,          0,        0,          0,
          37,         0,        0,         NaN,          0,        0,          0,
          38,         0,        0,         NaN,          0,        0,          0,
          39,         0,        0,         NaN,          0,        0,          0,
          40,         0,        0,         NaN,          0,        0,          0,
          41,         0,        0,         NaN,          0,        0,          0,
          42,         0,        0,         NaN,          0,        0,          0,
          43,         0,        0,         NaN,          0,        0,          0,
          44,         0,        0,         NaN,          0,        0,          0,
          45,         0,        0,         NaN,          0,        0,          0,
          46,         0,        0,         NaN,          0,        0,          0,
          47,         0,        0,         NaN,          0,        0,          0,
```

We can also study if a correlation exist between the the row labeling and the module configuration. For performing this analysis we always will need a labaling for rows and/or columns that group them in different sets. In this case we have as labeling the station number from which the bacteria and phages were extracted. Therefore what we will study is if there exist a correlation between the station location (geography) and the module configuration. We will use the same method that was used in Flores et al 2012. Given the labeling this method calculates the diversity index of the labeling inside each module and compare it with random permutations of the labeling across the matrix.

```
bp.testmodule.TestDiversityRows(1000); %Using the labeling of bp and 1000 random p
bp.testmodule.TestDiversityColumns(1000,moebus.phage_stations,@Diversity.SHANNON_I
bp.printer.PrintColumnModuleDiversity(); %Print the information of column diversit
```

```
        Diversity index:       Diversity.SHANNON_INDEX
        Random permutations:                      1000
        Module,index value, zscore,percent
             1,      2.4873,-1.8497,    3.8
             2,      2.2426,   -5.66,      0
             3,      1.9722,-1.5198,    4.9
             4,      2.0624,-5.6101,      0
             5,       1.398,-8.7294,      0
             6,      1.5942,-4.0556,    0.1
             7,      1.6094,0.55333,   26.1
             8,      1.0042,-5.8342,    0.1
             9,      1.4942,-2.8817,    0.8
```

```
    10,    0.79631,-6.8857,      0
    11,     1.7678,-2.8796,    0.6
    12,     1.3322, -1.319,    2.5
    13,      1.677,-2.4092,    1.3
    14,     1.0397,-1.8811,    1.2
    15,     1.0986,0.30199,    8.7
    16,          0,    NaN,      0
    17,          0,    NaN,      0
    18,          0,    NaN,      0
    19,    0.63651,-3.4851,      0
    20,          0,    NaN,      0
    21,          0,    NaN,      0
    22,          0,-5.7835,      0
    23,    0.69315,0.15673,    2.4
    24,          0,    NaN,      0
    25,          0,-6.3739,      0
    26,          0,    NaN,      0
    27,          0,    NaN,      0
    28,          0,    NaN,      0
    29,          0,    NaN,      0
    30,          0,    NaN,      0
    31,          0,    NaN,      0
    32,          0,    NaN,      0
    33,          0,    NaN,      0
    34,          0,    NaN,      0
    35,          0,    NaN,      0
    36,          0,    NaN,      0
    37,          0,    NaN,      0
    38,          0,    NaN,      0
    39,          0,    NaN,      0
    40,          0,    NaN,      0
    41,          0,    NaN,      0
    42,          0,    NaN,      0
    43,          0,    NaN,      0
    44,          0,    NaN,      0
    45,          0,    NaN,      0
    46,          0,    NaN,      0
    47,          0,-5.8889,      0
```

Published with MATLAB® 7.13