

1

2 **Supplementary Information for**

3 **Comparing meso-scale maps given by community detection algorithms on the Emmons**
4 **laboratory *Caenorhabditis elegans* connectome data**

5 **George Vernon**

6 **George Vernon**

7 **E-mail: george.vernon@warwick.ac.uk**

8 **This PDF file includes:**

9 Supplementary text

10 Supporting Information Text

11 **Subhead.** This SI provides the code used to generate and manipulate stochastic block models on the *C. elegans* connectome.
12 Some data must be imported first for the script to run properly.

13 Codes

14 **ERMM R script.** This script performs 1,000 iterations at a time for the specified `qmin` and `qmax`.

```
15 library(mixer)
16 NeuronLabels <- read.table("NeuronLabels.csv", sep=",")
17 connectomesymbin <- read.table("StrucConnectomeSymBinLabels.csv", header = TRUE, sep = ",")
18 bestcriterion <- -100000
19
20 numsamples = 1000
21 mtrx <- matrix(nrow = 3, ncol = numsamples)
22 bestbestcriterion <- bestbestm$criterion
23
24 start_time <- Sys.time()
25 for (i in 1:numsamples) {
26   mtrx[1,i] <- i
27   setSeed(unclass(Sys.time()))
28   mixer(connectomesymbin, qmin=5, qmin=15, nbiter=80, fpnbiter=40) -> ERMMout
29   m <- getModel(ERMMout)
30   m$criterion -> criterion
31   if (criterion > bestcriterion) {
32     bestcriterion <- criterion
33     bestm <- m
34     bestERMMout <- ERMMout
35   }
36   mtrx[2,i] <- criterion
37   mtrx[3,i] <- m$q
38   print("Done")
39 }
40
41 if (bestcriterion > bestbestcriterion) {
42   bestbestcriterion <- bestcriterion
43   bestbestm <- bestm
44   bestbestERMMout <- bestERMMout
45 }
46
47 sortedmtrx <- mtrx[, order(mtrx[2,],decreasing=TRUE)]
48 write.table(sortedmtrx, "filename.txt")
49
50 end_time <- Sys.time()
51
52 time <- end_time - start_time
53
54 print(time)
```

55 **ERMM map script.** This script extracts the group mapping from the MixeR `getModel` function output.

```
56 # R script for finding highest tau for each node
57
58 groupmap <- matrix(nrow = 1, ncol = 300)
59
60 for (i in 1:300) {
61   groupmap[i] <- which.max(bestbesttaus[,i])
62 }
63
64 write.table(groupmap, "groupmap.txt")
65 \end{lstlisting}
66
67 \subsection*{MATLAB Louvain Block Model Script}
68
69 This script performs 20,000 iterations of the Louvain algorithm, takes the best one and produces a plot.
70
71 \begin{lstlisting}
72 GapJunctionsSym = GapJunctionsSym - diag(diag(GapJunctionsSym));
73 GapJunctionsSymBin = GapJunctionsSym ~= 0;
74
75 ConnectomeSymBin = GapJunctionsSymBin + ChemSynapsBinSym;
76 ConnectomeSymBin = ConnectomeSymBin ~= 0;
77
```

```

78 ConnectomeSymBin = double(ConnectomeSymBin);
79 ConnectomeSymBinLabels = [NeuronLabels; ConnectomeSymBin];
80
81 writematrix(ConnectomeSymBinLabels,'StrucConnectomeSymBinLabels.csv');
82 writematrix(NeuronLabels, 'NeuronLabels.csv');
83
84 Q = 0;
85 for i = 1:20000
86     [Wi, Qi] = community_louvain(ConnectomeSymBin);
87     if Qi > Q
88         W = Wi;
89     end
90 end
91
92 ConnectomeSymBinCom = [W.'; NeuronLabels; ConnectomeSymBin] ;
93
94 [ConnectomeSymBinComSort, index] = sortrows(ConnectomeSymBinCom.', [1,2]);
95
96 ConnectomeSymBinSort = ConnectomeSymBin(index,index);
97
98 D = diff(sortrows(W));
99 S = find(D>0); % index where each group begins
100 S = S - 0.5; %centering the ticks
101
102 L = size(W)
103 T = [0;S;L(1)] % grid for the labels
104 U = zeros(5,1);
105 for i = 1:5
106     U(i) = (T(i) + T(i+1))/2
107 end
108
109 spy(ConnectomeSymBinSort,'k',2)
110 xlabel('')
111 grid on
112 ax = gca;
113 xticks(S)
114 yticks(S)
115 xticklabels([])
116 yticklabels([])
117 ax.GridColor = '#7E2F8E'
118 ax.GridLineStyle = '-';
119 ax.GridAlpha = 0.5;
120 ax.Layer = 'top';
121 ax.LineWidth = 1;
122 i = 0
123 grouplabels = cellstr(string(1:5));
124 text(zeros(1,5) - 10, U, grouplabels,'Color','black','FontSize',18,'HorizontalAlignment', 'right');
125 text(U - 10, zeros(1,5) + 315, grouplabels,'Color','black','FontSize',18);

```

126 **MATLAB Spectral Algorithm Script.** This script performs 20,000 iterations of the Spectral algorithm, takes the best one and
127 produces a plot.

```

128 GapJunctionsSym = GapJunctions - diag(diag(GapJunctions));
129 GapJunctionsSymBin = GapJunctionsSym ~= 0;
130
131 ConnectomeSymBin = GapJunctionsSymBin + ChemSynapsBinSym;
132 ConnectomeSymBin = ConnectomeSymBin ~= 0;
133
134 ConnectomeSymBin = double(ConnectomeSymBin);
135
136 Q = 0;
137 for i = 1:20000
138     [Wi, Qi] = modularity_und(ConnectomeSymBin);
139     if Qi > Q
140         W = Wi;
141     end
142 end
143
144 %relabelling so my groups match between Spectral and Louvain
145
146 for i = 1:300
147     if W(i) == 1
148         W(i) = 4;
149     elseif W(i) == 2

```

```

150     W(i) = 3;
151     elseif W(i) == 3
152         W(i) = 1;
153     elseif W(i) == 4
154         W(i) = 2;
155     end
156 end
157
158 ConnectomeSymBinCom = [W.'; NeuronLabels; ConnectomeSymBin] ;
159
160 [ConnectomeSymBinComSort, index] = sortrows(ConnectomeSymBinCom.',[1,2]);
161
162 ConnectomeSymBinSort = ConnectomeSymBin(index,index);
163
164 D = diff(sortrows(W));
165 S = find(D>0); % index where each group begins
166 S = S - 0.5; %centering the ticks
167
168 L = size(W);
169 T = [0;S;L(1)]; % grid for the labels
170 U = zeros(5,1);
171 for i = 1:5
172     U(i) = (T(i) + T(i+1))/2;
173 end
174
175 spy(ConnectomeSymBinSort,'k',2)
176 xlabel('')
177 grid on
178 ax = gca;
179 xticks(S)
180 yticks(S)
181 xticklabels([])
182 yticklabels([])
183 ax.GridColor = '#7E2F8E';
184 ax.GridLineStyle = '-';
185 ax.GridAlpha = 0.5;
186 ax.Layer = 'top';
187 ax.LineWidth = 1;
188 grouplabels = cellstr(string(1:5));
189 text(zeros(1,5) - 10, U, grouplabels,'Color','black','FontSize',18,'HorizontalAlignment','right');
190 text(U - 10, zeros(1,5) + 315, grouplabels,'Color','black','FontSize',18);

```

191 **Tikz script for producing the spring diagram.** This script is executable with the LuaLaTeX compiler to generate spring plots
192 visualising the group sizes and connection probabilities.

```

193 \RequirePackage{luatex85}
194 \documentclass[tikz,border=5]{standalone}
195 \usetikzlibrary{graphs,graphdrawing}
196 \usegdlibrary{force,trees}
197 \usepackage{luacode}
198 \begin{luacode*}
199
200 linewidthparameter = 0.1
201
202 function parseNetwork(nodeWeights, adjacencyMatrix)
203     local i, j, n, v, w, str
204     n = 0
205     weights = {}
206     for str in string.gmatch(nodeWeights, "[^%s]+") do
207         w = tonumber(str)
208         tex.print("n" .. n + 1 .. "[minimum size=" .. w .. ", label=above:Group " .. n + 1 .. ", label=center: " .. w
209             .. " ]");
210         n = n + 1
211     end
212     i = 0
213     j = 0
214     for str in string.gmatch(adjacencyMatrix, "[^%s]+") do
215         v = tonumber(str)
216         if v > 0 then
217             if j > i then
218                 tex.print("n" .. i + 1 .. " --[line width=" .. (v / linewidthparameter) .. "]" .. "n" .. j + 1 .. ";")
219             end
220         end
221         j = (j + 1) % n

```

```

222         if j == 0 then i = i + 1 end
223     end
224 end
225 \end{luacode*}
226 \tikzgraphsset{%
227     node weights/.store in=\nodeweights,
228     adjacency matrix/.store in=\adjacencymatrix,
229     declare={network}{
230         [/utils/exec={%
231             \edef\networkspec{\directlua{parseNetwork("\nodeweights","\adjacencymatrix")}},
232             parse/.expanded=\networkspec]
233         }}
234 \begin{document}
235 \begin{tikzpicture}
236 \graph [spring electrical layout,
237     edges={draw=blue!40!cyan, shorten >=-1em, shorten <=-1em},
238     nodes={circle, fill=blue!30!cyan!50},
239     electric charge=15]{
240     network [
241         node weights={
242         20    26    29    25    32    28    42    18    31    37    12
243         },
244         adjacency matrix={
245         0.58 0    0    0    0    0    0    0    0    0    0
246         0    0.38 0    0.21 0    0    0    0    0    0    0
247         0    0    0.22 0    0    0    0    0    0    0    0.29
248         0    0.21 0    0.36 0.1 0    0    0.18 0    0.13 0.33
249         0    0    0    0.1 0.21 0    0    0.10 0    0.15 0.26
250         0    0    0    0    0    0.29 0    0.25 0    0    0.29
251         0    0    0    0    0    0    0.24 0.09 0    0    0.33
252         0    0    0    0.18 0.10 0.25 0.09 0.58 0    0    0.46
253         0    0    0    0    0    0    0    0    0.26 0.16 0.08
254         0    0    0    0.13 0.15 0    0    0    0.16 0.39 0.16
255         0    0    0.29 0.33 0.26 0.29 0.33 0.46 0.08 0.16 0.83
256         }];
257 };
258 \end{tikzpicture}
259 \end{document}
260
261 % https://tex.stackexchange.com/questions/362451/how-to-produce-this-matlab-social-network-graph-from-an-adjacency-
262 matrix-in-tikz This code modified from StackExchange and used with permission under the MIT license

```