**Assignment 2 (10%)**

**Course co-ordinator: Dr Marcus Claesson (m.claesson@ucc.ie)**

**Demonstrators: Joseph Pennycook**

**Student Name (ID): James Henry Hehir (120224791)**

For each question, **please record your answer in a Word document, and also what you typed on the command-line to get this answer. When finished, upload the document (with your name in the title) with your answers to Canvas under Assignments as MB6300 Assignment 2.**

**Deadline: Sun 14th March**

If you cannot meet this deadline you will have to fill in and submit the Late Submission Form (see Assignment folder on Canvas) and provide any supporting documentation (e.g. medical cert).

All students are expected to work individually on assignments; those found collaborating with others will receive a score of zero for their work.

Include relevant steps that you used to get your answer. If you include intermediate steps, you will obtain marks even if your final answer is incorrect. Code should be as efficient and automated as possible.

Do not use the edit() function in any parts of this assignment.

All R code should include comments briefly describing how it works, and what it is doing.

**Q.1 The file, assignment2.RData, in the Assignment 2 folder on Canvas is an R environment containing 4 undirected graphs, two pairs. These graphs are labelled, graph1, graph2, graph3 and graph4. Each pair contains the same number of nodes (genes) and edges (interactions); however, within each pair there is a graph from a real gene regulatory network and a graph where edges have been assigned randomly between nodes (see lecture notes). Remember to load all necessary libraries. (4)**

**(a)** Identify each pair and demonstrate, using a metric/method of your choice, which graph is real and which is random. Explain the logic behind your decision.

> min(nodes(graph1)) #getting the min of graph 1

[1] "1"

> max(nodes(graph1)) #getting the max of graph 1

[1] "999"

> min(nodes(graph2))

[1] "iw3pJUpdK1vimsNMpH93R3mS+u44932(rogid)"

> max(nodes(graph2))

[1] "YPR193C"

> min(nodes(graph3))

[1] "Q0080"

> max(nodes(graph3))

[1] "YPR191W"

> min(nodes(graph4))

[1] "1"

> max(nodes(graph4))

[1] "999"

Graph 2 and 3 are real as their nodes have strings of nodes while graph 1 and 4 have only integers for their nodes which make them the random graphs.

**(b)** How many nodes are in the second largest connected component for each of the real graphs? Plot these connected components. If there is more than one connected component that is the second largest size plot all second largest components (for example if there are 3 connected components in the graph that have 8 nodes, and 8 is the second largest, plot all 3 of these connected components).

> con\_comp\_graph2 <- connComp(graph2) #returns a list of connected components

> sum\_graph2 <- (lapply(con\_comp\_graph2, function(x)

+ sum(lengths(x)))) #find the nodes with the highest connected component

> which(sum\_graph2 >7)

[1] 1

> which(sum\_graph2 == 7)

[1] 2 13 25 79 95

> con\_comp\_graph3 <- connComp(graph3) #returns a list of connected components

> sum\_graph3 <- (lapply(con\_comp\_graph3, function(x)

+ sum(lengths(x))))

> which(sum\_graph3 >10)

[1] 1

> which(sum\_graph3 == 10)

[1] 27

library(Rgraphviz)

sum\_graph2\_plot <- RBGL::connectedComp(graph2)

graph2\_plot <- subGraph(sum\_graph2\_plot[[7]], graph2) #makes subgraph with nodes you want

Rgraphviz::layoutGraph(graph2\_plot) #this layouts graph objects

Rgraphviz::renderGraph(Rgraphviz::layoutGraph(graph2\_plot, layoutType = "neato")) #this renderes the laid out graph from previous line

sum\_graph3\_plot <- RBGL::connectedComp(graph3)

graph3\_plot <- subGraph(sum\_graph3\_plot[[10]], graph3) #makes subgraph with nodes you want

Rgraphviz::layoutGraph(graph3\_plot) ) #this layouts graph objects

Rgraphviz::renderGraph(Rgraphviz::layoutGraph(graph3\_plot, layoutType = "neato")) #this renderes the laid out graph from previous line

**Plot of Graph 2**

Diagram

Description automatically generated

**Plot of Graph 3**

Diagram

Description automatically generated

**(c)** Which node has the highest degree in each real graph? List the nodes that have edges connecting them to this node. Explain this node in a biological context.

> graph2\_degs <- graph::degree(graph2)#creates a graph from a graph

> which.max(graph2\_degs) #finds the max

YCL032W

100

> graph3\_degs <- graph::degree(graph3) #creates a graph from a graph

> which.max(graph3\_degs) #finds the max

YCR057C

210

> graph::adj(graph2, "YCL032W") #this is a method for finding the adjacent list of the selected node

$YCL032W

[1] "YCR082W" "YDL016C" "YDL239C" "YDR032C"

[5] "YDR308C" "YDR309C" "YDR386W" "YDR416W"

[9] "YEL051W" "YER040W" "YER047C" "YGL233W"

[13] "YHR061C" "YJR093C" "YKL061W" "YKR020W"

[17] "YLL049W" "YLR223C" "YLR362W" "YMR270C"

[21] "YNL086W" "YOL043C" "YOL123W" "YOR036W"

[25] "YOR270C" "YPL120W" "YPR051W" "YPR182W"

[29] "YPR193C" "YAL047C"

> graph::adj(graph3, "YCR057C") #this is a method for finding the adjacent list of the selected node

$YCR057C

[1] "YCR057C" "YDL014W" "YDL148C" "YDL213C"

[5] "YDR299W" "YDR324C" "YDR365C" "YDR382W"

[9] "YDR449C" "YEL050C" "YER082C" "YGL011C"

[13] "YGL120C" "YGL171W" "YGR090W" "YGR128C"

[17] "YGR135W" "YGR145W" "YHR148W" "YHR169W"

[21] "YHR196W" "YJL033W" "YJL069C" "YJL109C"

[25] "YJR002W" "YKL099C" "YKR060W" "YLL011W"

[29] "YLR129W" "YLR175W" "YLR186W" "YLR197W"

[33] "YLR222C" "YLR409C" "YML130C" "YMR093W"

[37] "YMR128W" "YMR300C" "YNL075W" "YNL132W"

[41] "YNR043W" "YNR054C" "YOL038W" "YOR078W"

[45] "YOR310C" "YPL126W" "YPL217C" "YPR137W"

[49] "YPR144C" "YBL003C" "YBL004W" "YBR009C"

[53] "YBR247C" "YCL059C"

High degree nodes in a protein interaction network are enriched in essential proteins. The nodes selected here have the highest number of nodes which means they have the highest degree. This also means they have a high centrality.

**(d)** How many nodes have a degree of one in each real graph? Explain these nodes in a biological context.

> con\_comp\_graph2 <- connComp(graph2) # get connected components

> sum\_graph2 <- (lapply(con\_comp\_graph2, function(x)

+ sum(lengths(x))))

> length\_graph2 <- which(sum\_graph2 == 1) #gets nodes equal to 1

> length\_graph2

[1] 17 77 80 107 111 120 124 126 135 141

[11] 142 149 154

> length(length\_graph2) # gets the number of nodes

[1] 13

> con\_comp\_graph3 <- connComp(graph3)

> sum\_graph3 <- (lapply(con\_comp\_graph3, function(x)

+ sum(lengths(x))))

> length\_graph3 <- which(sum\_graph3 == 1) #gets nodes equal to 1

> length\_graph3

[1] 2 4 5 6 7 8 9 12 13 14

[11] 15 16 17 18 19 20 21 22 23 26

[21] 28 29 30 32 33 34 35 37 38 39

[31] 40 42 43 45 46 49 50 51 52 53

[41] 54 56 57 58 61 62 63 64 65 67

[51] 68 69 70 71 72 73 75 76 77 81

[61] 83 85 86 87 89 90 91 92 94 95

[71] 97 99 100 103 104 107 108 109 110 111

[81] 112 113 115 116 118 120 121 122 123 124

[91] 126 128 130 131 132 133 134 135 136 137

[101] 138 139 140 141 143 144 145 146 147 148

[111] 149

> length(length\_graph3) # gets the number of nodes

[1] 111

All nodes with a degree of one have a low centrality. This means they are less important due to the lack of degree.

**Q.2 Load the igf dataset from the BoolNet library into your R environment. The transition functions in igf resemble the cellcycle dataset (Practical 5) in terms of their structure, but there is a difference. Some transition functions have either -2 or -3 in square brackets after certain genes. When answering questions, remember that the cellcycle dataset transitions from t to t+1 in a deterministic manner that is reflected by the transition function for each gene. The igf dataset follows the same deterministic structure (i.e. earlier states completely determine later states). (4)**

**(a)** How many genes are in the igf dataset? List the genes. How many attractors are there? Display the attractors.

> length(igf$genes) # gets how many genes

[1] 6

> igf$genes # lists the genes

[1] "IGF" "IRS" "PI3K" "Akt" "mTORC1" "mTORC2"

> igf\_attractors <- getAttractors(igf) # get attractors

> length(igf\_attractors$attractors) # gets the number of

[1] 2

> igf\_attractors$attractors

[[1]]

IGF IRS PI3K Akt mTORC1 mTORC2

1 0 0 0 0 0 0

[[2]]

IGF IRS PI3K Akt mTORC1 mTORC2

1 1 0 0 0 0 0

2 1 1 0 0 0 0

3 1 1 1 0 0 0

4 1 1 1 1 0 1

5 1 1 1 1 0 1

6 1 1 1 1 0 0

7 1 1 1 1 1 0

8 1 1 1 1 1 0

9 1 0 1 1 1 0

10 1 0 0 1 1 0

11 1 0 0 0 1 0

12 1 0 0 0 1 0

13 1 0 0 0 0 0

14 1 0 0 0 0 0

**(b)** How many states does each attractor have in the igf dataset? Explain the difference between attractor 1 and attractor 2 in terms of how the model works/runs? For this please explain the general difference between the two in terms of states and cycling.

> igf\_attractors

Simulation of a symbolic Boolean network

Graph containing 331 state transitions (print with graph=TRUE to show them)

2 Attractors:

Attractor 1 is a simple attractor consisting of **1 state(s):**

|--<-----|

V |

000000 |

V |

|-->-----|

Genes are encoded in the following order: IGF IRS PI3K Akt mTORC1 mTORC2

Attractor 2 is a simple attractor consisting of **14 state(s):**

|--<-----|

V |

100000 |

110000 |

111000 |

111101 |

111101 |

111100 |

111110 |

111110 |

101110 |

100110 |

100010 |

100010 |

100000 |

100000 |

V |

|-->-----|

Genes are encoded in the following order: IGF IRS PI3K Akt mTORC1 mTORC2

In this example we have two attractors. Attractor 1 only has 1 while Attractor 2 has 14 states. Attractor one is a simple one while Attractor 2 is a complex attractor. Attractor 2 usually have more than one possible transition for each state in an asynchronous network. It is a set of states which lead to another state in that set and a state in the set can be reached from all other states in the set.

Attractors can have states which is a point in the attractor. Similarly if there is a set of states you can have a cycle of states. This can be seen in Attractor 2.

**(c)** Explain what -2 and -3 represent in square brackets after certain genes in the transition functions. Describe the transition function for the IRS gene in words.

The operators represented in square brackets after certain genes can incorporate time ranges. They define a time delay that be can be used for time specifics. This is useful as a transition function can also depend on the states of genes at different time points.

The transition function for the IRS gene is the role of transmitting signals form the insulin receptors to intracellular pathways.

**(d)** Display the basin of attraction for attractor 2 (i.e. all the states that lead to attractor 2 after one state transition).

> getBasinOfAttraction <- function(attractorInfo,attractorNo)

+ {

+ stopifnot(inherits(attractorInfo,"AttractorInfo") || inherits(attractorInfo,"SymbolicSimulation"))

+

+ if (missing(attractorNo) || attractorNo <= 0 || attractorNo > length(attractorInfo$attractors))

+ stop("Please provide a valid attractor number!")

+

+ table <- getTransitionTable(attractorInfo)

+ return(table[which(table$attractorAssignment == attractorNo),,drop=FALSE])

+ }

> attractors <- getAttractors(igf)

> print(getBasinOfAttraction(attractors, 2))

State Next state Attr. basin

100000 => 100000 2

100000 => 110000 2

110000 => 111000 2

111000 => 111101 2

111101 => 111101 2

111101 => 111100 2

111100 => 111110 2

111110 => 111110 2

111110 => 101110 2

101110 => 100110 2

100110 => 100010 2

100010 => 100010 2

100010 => 100000 2

Genes are encoded in the following order: IGF IRS PI3K Akt mTORC1 mTORC2

**Q.3 This question will draw on your knowledge from Assignment 1 and test your ability to incorporate specific functions from R network packages into functions of your own. Remember that a good attempt at the answer will get you a lot of the marks. (2)**

**(a)** Write a function that takes a graph (like those from Q.1) and outputs a matrix of one column and six rows with the following information: minimum degree, median degree, mean degree, maximum degree, number of connected components and number of nodes (genes) in the largest connected component. Only values are necessary, not gene names. Output the single column and six rows with appropriately labelled column and row names.

graph\_info <- function(graph){

info <- igraph.from.graphNEL(graph, unlist.attrs = TRUE) #converts graphNEL objects from graph to igraph

degree\_graph <- degree(info) #gives the info

output\_mtx1 <- matrix(, ncol = 1, nrow = 6) # create blank matrix

con\_comp\_graph <- connComp(graph) #get connected components

sum\_comp\_graph <- (lapply(con\_comp\_graph,function(x) sum(lengths(x))))

node\_count <- max(unlist(sum\_comp\_graph)) # get node count in largest component

rownames(output\_mtx1) <- c("Min Deg", "Median Deg", "Mean Deg", "Max Deg", "Connected Components", "Nodes") # rownames

colnames(output\_mtx1) <- ("values") #col names

output\_mtx1[1, 1] <- min(degree\_graph) #place min in matrix

output\_mtx1[2, 1] <- median(degree\_graph) #place median in matrix

output\_mtx1[3, 1] <- mean(degree\_graph) #place mean in matrix

output\_mtx1[4, 1] <- max(degree\_graph) #place max in matrix

output\_mtx1[5, 1] <- no.clusters(info) #place clusters in matrix

output\_mtx1[6, 1] <- node\_count #place nodes in matrix

return(output\_mtx1) #return the matrix

}

**OUTPUT:**

> graph\_info(graph2)

values

Min Deg 1.000000

Median Deg 1.000000

Mean Deg 2.165663

Max Deg 30.000000

Connected Components 156.000000

Nodes 921.000000

**(b)** Write a function that takes a Boolean network (like cellcycle or igf) and outputs the basic network information (what you get when you type the network name, e.g. cellcycle, on the command line and hit ‘return’) along with a matrix of one column and two rows with values for the number of genes and the number of attractors. Only values are necessary, not gene names. Output the single column and two rows with appropriately labelled column and row names.

network\_info <- function(network){

info <- print(network) #print the info of network

output\_mtx <- matrix(, ncol = 1, nrow = 2) #empty matrix

rownames(output\_mtx) <- c("Genes", "Attractors") # name the rows

colnames(output\_mtx) <- ("Values") # name the columns

output\_mtx[1, 1] <- length(network$genes) #place in matrix the number of genes

output\_mtx[2, 1] <- length(getAttractors(network)) #no of attractors

return(output\_mtx)

}

**OUTPUT:**

> network\_info(igf)

Symbolic representation of a Boolean network

Transition functions:

IGF = IGF

IRS = (IGF[-2] & !mTORC1[-3] & !mTORC1[-2])

PI3K = IRS

Akt = (PI3K | mTORC2)

mTORC1 = (Akt[-3] & Akt[-2])

mTORC2 = (!(Akt[-2] | mTORC1[-2]) & PI3K)

Values

Genes 6

Attractors 2