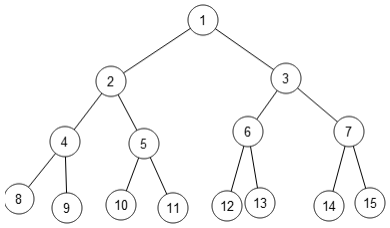
ST5225 Assignment

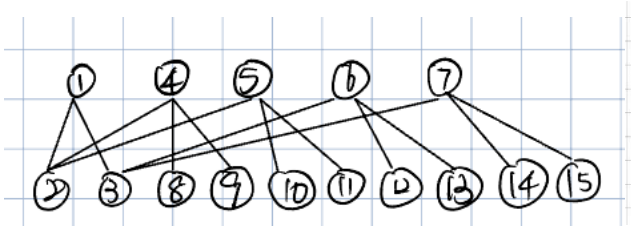
Question 1:

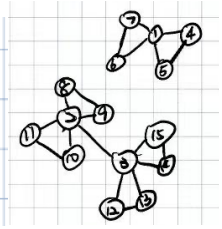
(a) Can you express the graph in Figure 1 as a bipartite graph? That is divide the nodes into 2 groups such that all edges are between the two groups.




```
> A = matrix(nrow = 15, ncol = 15, data = 0)
> rownames(A) = colnames(A) = c("x1", "x2", "x3", "x4", "x5", "x6", "x7",
+                               "x8", "x9", "x10", "x11", "x12", "x13", "x14", "x15")
> A[1,2] = A[1,3] = A[2,4] = A[2,5] = A[4,8] = A[4,9] = A[5,10] = A[5,11] = 1
> A[3,6] = A[3,7] = A[6,12] = A[6,13] = A[7,14] = A[7,15] = 1
> A = A + t(A)
> A
    x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15
x1   0  1  1  0  0  0  0  0  0   0   0   0   0   0   0
x2   1  0  0  1  1  0  0  0  0   0   0   0   0   0   0
x3   1  0  0  0  0  1  1  0  0   0   0   0   0   0   0
x4   0  1  0  0  0  0  0  1  1   0   0   0   0   0   0
x5   0  1  0  0  0  0  0  0  0   1   1   0   0   0   0
x6   0  0  1  0  0  0  0  0  0   0   0   1   1   0   0
x7   0  0  1  0  0  0  0  0  0   0   0   0   0   1   1
x8   0  0  0  1  0  0  0  0  0   0   0   0   0   0   0
x9   0  0  0  1  0  0  0  0  0   0   0   0   0   0   0
x10  0  0  0  0  1  0  0  0  0   0   0   0   0   0   0
x11  0  0  0  0  1  0  0  0  0   0   0   0   0   0   0
x12  0  0  0  0  0  1  0  0  0   0   0   0   0   0   0
x13  0  0  0  0  0  1  0  0  0   0   0   0   0   0   0
x14  0  0  0  0  0  0  1  0  0   0   0   0   0   0   0
x15  0  0  0  0  0  0  1  0  0   0   0   0   0   0   0
```

```
> B = A%*%A
> A = matrix(nrow = 15, ncol = 15, data = 0)
> B
    x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15
x1   2  0  0  1  1  1  1  0  0   0   0   0   0   0   0
x2   0  3  1  0  0  0  0  1  1   1   1   0   0   0   0
x3   0  1  3  0  0  0  0  0  0   0   0   1   1   1   1
x4   1  0  0  3  1  0  0  0  0   0   0   0   0   0   0
x5   1  0  0  1  3  0  0  0  0   0   0   0   0   0   0
x6   1  0  0  0  0  3  1  0  0   0   0   0   0   0   0
x7   1  0  0  0  0  1  3  0  0   0   0   0   0   0   0
x8   0  1  0  0  0  0  0  1  1   0   0   0   0   0   0
x9   0  1  0  0  0  0  0  1  1   0   0   0   0   0   0
x10  0  1  0  0  0  0  0  0  0   1   1   0   0   0   0
x11  0  1  0  0  0  0  0  0  0   1   1   0   0   0   0
x12  0  0  1  0  0  0  0  0  0   0   0   1   1   0   0
x13  0  0  1  0  0  0  0  0  0   0   0   1   1   0   0
x14  0  0  1  0  0  0  0  0  0   0   0   0   0   1   1
x15  0  0  1  0  0  0  0  0  0   0   0   0   0   1   1
```





Group A: x1, x4, x5, x6, x7
Group B: x2, x3, x8, x9, x10, x11, x12, x13, x14

So, we can express the graph in Figure 1 as a bipartite graph, dividing the nodes into 2 groups such that all edges are between the two groups. Accordingly, all edges connect nodes from different groups, satisfying the property of a bipartite graph.

(b) Compute the modularity of your bipartite graph.

Degree Sequence $k = (k_1, k_2, \cdots\cdots, k_{16}) = (2, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1)$

$J = 2m = 28$

Let $x_u^A = 1$ if node u is from group $A: \{\pi_1, \pi_4, \pi_5, \pi_6, \pi_7\}$

if not let $x_u^A = 0$

Let y^A and z^A be the vector created from I_2 and x_u^A

$$Q_A = \text{cov}(y^A, z^A)$$

$$= \frac{1}{J}\sum_{j=1}^{J} y_j^A z_j^A - \bar{y}^A z^A$$

$$= \frac{1}{2m}\sum_{u=1}^{n}\sum_{v=1}^{n} a_{uv} x_u^A x_v^A - (\bar{\pi}_E^A)^2$$

$$= \frac{1}{28}[0] - \frac{1}{4} = -\frac{1}{4}$$

$$\bar{\pi}_E^A = \frac{1}{2m}\sum_{u=1}^{n} k_u \pi_u^A$$

$$= \frac{1}{28}\cdot[2+3+3+3+3] = \frac{1}{2}$$

$$Q_B = \text{cov}(y^B, z^B)$$

$$= \frac{1}{J}\sum_{j=1}^{J} y_j^B z_j^B - \bar{y}^B z$$

$$= \frac{1}{2m}\sum_{u=1}^{n}\sum_{v=1}^{n} a_{uv} x_u^B x_v^B - (\bar{\pi}_E^B)^2$$

$$= \frac{1}{28}[0] - \frac{1}{4} = -\frac{1}{4}$$

$$\bar{\pi}_E^B = \frac{1}{2m}\sum_{u=1}^{n} k_u \pi_u^B$$

$$= \frac{1}{28}\cdot[3+3+1\times 8] = \frac{1}{2}$$

$Q = Q_A + Q_B = -\frac{2}{4} = -\frac{1}{2}$

So the network is disassortative mixing for bipartite graph with modularity $Q = -\frac{1}{2}$

(c) Is it true that all trees can be expressed as a bipartite graph? Is the expression unique? That is can there be two distinct ways of dividing the nodes into 2 groups? Please elaborate.

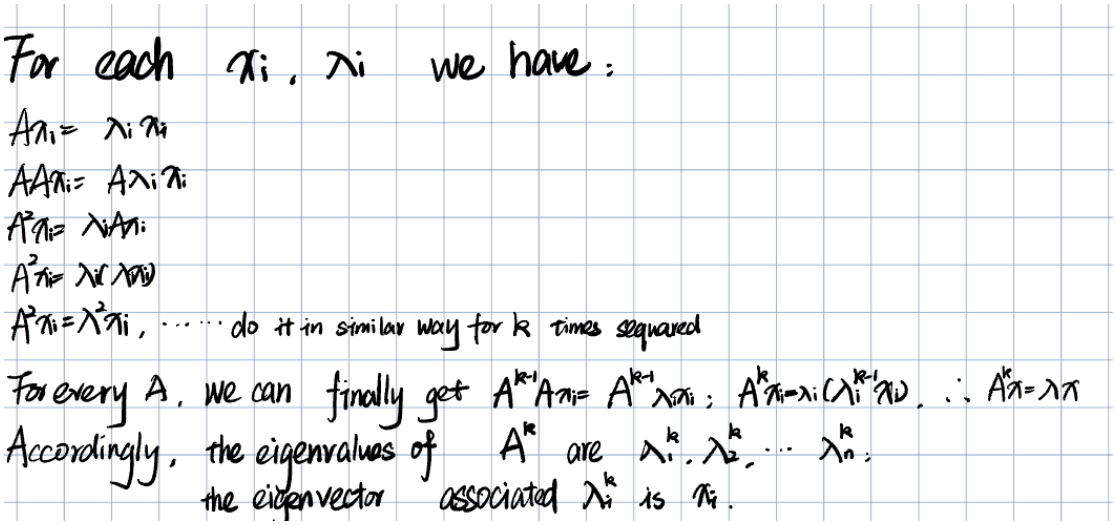
Yes, all trees can be expressed as a bipartite graph.
According to the definition, a tree is a connected, undirected network containing no loops. In a tree, the number of edges $m = n - 1$. As there are no cycles in a tree, we can select any node at the root of the tree and then place nodes at even levels in one group, and the rest in another group. That is, take any vertex in the tree that is an even number of edges away put it in group A, and that is an odd number of edges away put it in group B.
If two vertices v_1, and v_2 in A are connected by edge e, we can make a loop through the path from v_0 to v_1, v_0 to v_2, and edge e. However, for a tree, there should be no loops according to the definition. It is contradicting. A similar situation for group B.

Consequently, we can always divide the nodes into two groups such that all edges are connected by nodes from different groups.

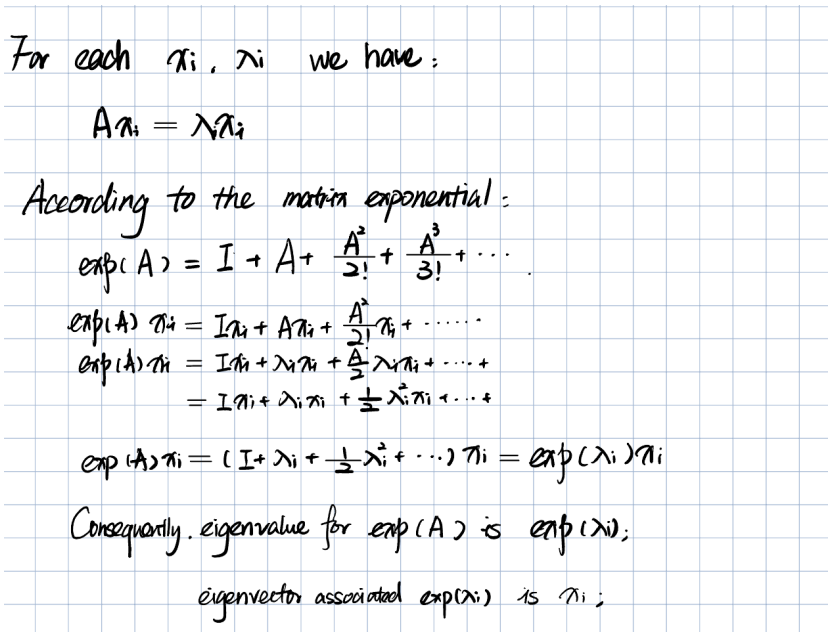
Question 2:

Let A be an n×n symmetric matrix. Let λ1, . . . , λn be its eigenvalues and let xi be the eigenvector associated with λi.

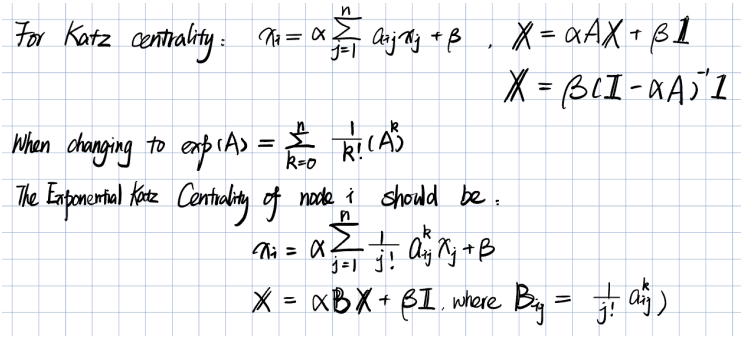
(a) Let k be a non-negative integer. Show that the eigenvalues of A^k are λ_1^k, . . . , λ_n^k and that the eigenvector associated with λ_i^k is x_i.



(b) What are the eigenvalues and eigenvectors of exp(A)? Hint: This is simpler than it looks. The answer is similar to that of Question 2(a).



(c) A graph researcher would like to define a new measure of node centrality using the matrix exponential. Can you help him? It is similar to Katz centrality. Provide some explanation of what your node centrality is measuring.



This Exponential Katz Centrality could also measure the centrality of a node by considering all possible paths in the network. Moreover, it could provide more weight to the shorter paths in the graph, while it would provide less weight to longer paths in the graph. Accordingly, it could provide a measure of the influence of a node based on the direct connection, intermediate node neighbors, and extended paths in detail by measuring the length, taking the structures and path length of the network into account. Nodes with high centrality tends to have a higher influence on various length path in the graph, and vice versa.

Question 3:

The matrix exponential can be used in link prediction. Consider a graph generated using the codes below. Let A be the adjacency matrix and remove 48 of the edges randomly to form B. We want to predict the missing edges using B.

(a) Apply the matrix exponential exp(αA) instead of A 2for link prediction. Do note that you need to download the package expm to do this. Do not type exp(A) using the R software directly as you will simply get exp(a[i,j]) for the entry in the ith row and jth column. Consider two values of α and decide which α is the better one.

```
> ########## (a)
> library(expm)
> alphas = c(0.4, 0.65)
> for (alpha in alphas) {
+   D = expm::expm(alpha * A)
+   pred_a = D * (1 - B) * (1-diag(100))
+   sort(-pred_a)[1:96]
+   correct_predictions = sum((pred_a > 5) * A)/2
+   cat("Alpha =", alpha, "Correct Predictions =", correct_predictions, "\n")
+ }
Alpha = 0.4 Correct Predictions = 3
Alpha = 0.65 Correct Predictions = 48
```

According to the number of correct predictions, alpha = 0.65 is the better one.

(b) Explain how the method above can be made more complete by using cross-validation. That is using cross-validation to choose the best value of α.

First, split the dataset into k folds. For each fold, we can use k-1 folds for training and the remaining one for testing. Then, perform the previous progress, that is, computing exp(αA), and then predict the missing edges to compare with the actual edges in the test set to evaluate the performance of the metric. Accordingly, we can get the average performance metrics over all folds for each α. The better the average performance metrics, the better α value is, that is, this α could result in the best average performance metric.

(c) Repeat the exercise in Question 3(a) but now using preferential attachment. That is we score a node pair (u, v) by k u k v , where k u is the degree of node u in the graph with missing edges.

```
> ################ (c)
> node_degrees = colSums(B)
> s = matrix(nrow=100, ncol=100, data=0)
> for (i in 1:100) {
+   for (j in 1:100) {
+     s[i, j] = node_degrees[i] * node_degrees[j]
+   }
+ }
> pred_c = s * (1 - A) * (1-diag(100))
> sort(-pred_c)[1:39]
 [1] -225 -225 -210 -210 -210 -210 -210 -210 -210 -210
[11] -196 -196 -196 -196 -196 -196 -196 -196 -196 -196
[21] -196 -196 -196 -196 -195 -195 -195 -195 -195 -195
[31] -195 -195 -195 -195 -195 -195 -182 -182 -182
> correct_predictions_c = sum((pred_c > 5) * A)/2
> correct_predictions
[1] 48
```

Question 4:

In the codes below we constructed a graph using a block model of three blocks with 30 nodes in each block. In addition we have also constructed a network object A2.network containing the given graph and node attributes "xval" with values 1–3 depicting which group the nodes belong to.

(a) Compute the transitivity and density of the graph.

```
> ############################ (a)
> library(igraph)
> graph = graph_from_adjacency_matrix(A2, mode = "undirected")
> transitivity_value = transitivity(graph)
> print(transitivity_value)
[1] 0.316307
> density_value = edge_density(graph)
> print(paste("Density:", density_value))
[1] "Density: 0.229213483146067"
> # Method 2: alculate directly
> n = nrow(A2)
> num_edges = sum(A2) / 2
> total_possible_edges = n * (n - 1) / 2
> density = num_edges / total_possible_edges
> print(density)
[1] 0.2292135
```

(b) Is the transitivity of this graph significant against a random graph with the same degree sequence? Generate 999 random graphs with the same degree sequence and find a one-sided Monte Carlo p-value.

```
> ######### (b)
> num_simulations = 999
> n = nrow(A2)
> generate_random_graph = function(A2) {
+   degree_sequence = degree(graph_from_adjacency_matrix(A2, mode = "undirected"))
+   random_graph = sample_degseq(degree_sequence, method = "vl")
+   return(random_graph)
+ }
> num_greater_transitivity = 0
> for (i in 1:num_simulations) {
+   random_graph = generate_random_graph(A2)
+   random_transitivity = transitivity(random_graph)
+   if (random_transitivity >= transitivity_value) {
+     num_greater_transitivity = num_greater_transitivity + 1
+   }
+ }
> monte_carlo_p_value = num_greater_transitivity / num_simulations
> monte_carlo_p_value
[1] 0
```

Accordingly, the one-sided Monte Carlo p-value is smaller than 0.05, which suggests that the observed result is statistically significant. The transitivity of the graph is significant against a random graph with degree sequence.

(c) Assume that the node attributes xi are known to the user. Apply an appropriate logistic regression model and make the necessary conclusions.

```
> A2.indep=ergm(A2.network~edges+match("xval"))
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Stopping at the initial estimate.
Evaluating log-likelihood at the estimate.
> summary(A2.indep)
Call:
ergm(formula = A2.network ~ edges + match("xval"))

Maximum Likelihood Results:

              Estimate Std. Error MCMC % z value Pr(>|z|)
edges          -2.20548    0.06435      0  -34.27   <1e-04 ***
nodematch.xval  2.19782    0.08489      0   25.89   <1e-04 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

     Null Deviance: 5552  on 4005  degrees of freedom
 Residual Deviance: 3556  on 4003  degrees of freedom

AIC: 3560  BIC: 3572  (Smaller is better. MC Std. Err. = 0)
```

In this question, we are investigating a block model of three blocks with 30 nodes in each block, that is, it can be divided into 3 groups, and the probability that $(u, v) \in E$ depends only on which groups u and v belong to.

We should be interested in how the cooperation of A2 may depend on x_i. The maximum likelihood estimation of β can be performed using logistic regression. The model aims to capture the patterns of connections in the network based on these variables and their assortative tendencies. The graph models allow for dependent edges but they also encompass independent edges as a special case. The advantage of using this package is that we can work directly with the adjacency matrix of the network and the covariates of the nodes. There is no need to construct a separate dataset containing N observations and the covariates in h_{uv} for each node pair (u, v).

According to the table for the result, we can see $\widehat{\beta_1} = 2.19782$. All the p-values are small. This indicates that all the effects β are non-zero. Since $exp(\hat{\beta}) = 9.005$, so we can say that for every increase in x_i, the odds of an interaction with any others increase by a factor of 9.005.

(d) Assume that the node attribute information is not available. Apply an ERGM with GWESP and GWD counts. Conclude which terms are significant in your model.

```
> ######### (d)
> A2.ergm = ergm(A2.network ~ edges + gwesp(0.5, fixed = TRUE))
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Starting Monte Carlo maximum likelihood estimation (MCMLE):
Iteration 1 of at most 60:
Optimizing with step length 0.8848.
The log-likelihood improved by 2.3151.
Estimating equations are not within tolerance region.
Iteration 2 of at most 60:
Optimizing with step length 1.0000.
The log-likelihood improved by 0.3799.
Estimating equations are not within tolerance region.
Iteration 3 of at most 60:
Optimizing with step length 1.0000.
The log-likelihood improved by 0.0734.
Convergence test p-value: 0.1809. Not converged with 99% confidence; increasing sample size.
Iteration 4 of at most 60:
Optimizing with step length 1.0000.
The log-likelihood improved by 0.0281.
Convergence test p-value: 0.0036. Converged with 99% confidence.
Finished MCMLE.
Evaluating log-likelihood at the estimate. Fitting the dyad-independent submodel...
Bridging between the dyad-independent submodel and the full model...
Setting up bridge sampling...
```

```
The log-likelihood improved by 0.0281.
Convergence test p-value: 0.0036. Converged with 99% confidence.
Finished MCMLE.
Evaluating log-likelihood at the estimate. Fitting the dyad-independent submodel...
Bridging between the dyad-independent submodel and the full model...
Setting up bridge sampling...
Using 16 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 .
Bridging finished.
This model was fit using MCMC.  To examine model diagnostics and check for degeneracy, use the
mcmc.diagnostics() function.
> summary(A2.ergm)
Call:
ergm(formula = A2.network ~ edges + gwesp(0.5, fixed = TRUE))

Monte Carlo Maximum Likelihood Results:

                Estimate Std. Error MCMC % z value Pr(>|z|)
edges           -1.9372     0.3223      0  -6.011   <1e-04 ***
gwesp.fixed.0.5  0.3549     0.1584      0   2.241    0.025 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 5552  on 4005  degrees of freedom
 Residual Deviance: 4307  on 4003  degrees of freedom

AIC: 4311  BIC: 4324  (Smaller is better. MC Std. Err. = 0.2703)
```

According to the table for the result, we can see $\widehat{\beta_1} = 0.3549$. The p-values are small. This indicates that the effects β are non-zero. Since $exp(\hat{\beta}) = 1.426038$, so we can say that for every increase in x_i, the odds of an interaction with any others increase by a factor of 1.426038.

(e) Next apply ERGM but with the node information available. Are the GWESP and GWD counts significant?

```
> ######### (e)
> summary(A2.network~edges+gwesp(1.1,fixed=TRUE)+gwdegree(2,fixed=TRUE)+triangle+kstar(2))
         edges gwesp.fixed.1.1   gwdeg.fixed.2        triangle          kstar2
      918.0000       2337.5111        626.5751       1930.0000      18305.0000
> N=90*89/2
> exp(1.1)*(1-(1-exp(-1.1))^88)*N
[1] 12031.68
> exp(2)*(1-(1-exp(-2))^89)*90
[1] 665.0135
> alpha=exp(2)*(1-(1-exp(-2))^c(1:6))
> alpha
[1] 1.000000 1.864665 2.612310 3.258772 3.817745 4.301070
> inc=alpha[2:6]-alpha[1:5]
> inc
[1] 0.8646647 0.7476451 0.6464623 0.5589732 0.4833244
> A2.dep=formula(A2.network~edges+gwesp(1.1,fixed=TRUE)+gwdegree(2,fixed=TRUE)+nodematch("xval"))
> summary(A2.dep)
         edges gwesp.fixed.1.1   gwdeg.fixed.2  nodematch.xval
      918.0000       2337.5111        626.5751        650.0000
> set.seed(2)
> laz.ergm=ergm(A2.dep)
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Finished MCMLE.
Evaluating log-likelihood at the estimate. Fitting the dyad-independent submodel...
Bridging between the dyad-independent submodel and the full model...
Setting up bridge sampling...
Using 16 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 .
Bridging finished.
This model was fit using MCMC.  To examine model diagnostics and check for degeneracy, use the
mcmc.diagnostics() function.
> summary(laz.ergm)
Call:
ergm(formula = A2.dep)

Monte Carlo Maximum Likelihood Results:

               Estimate Std. Error MCMC % z value Pr(>|z|)
edges          -2.44934    0.34995      0  -6.999   <1e-04 ***
gwesp.fixed.1.1  0.02488    0.06796      0   0.366    0.714
gwdeg.fixed.2    1.36944    1.61623      0   0.847    0.397
nodematch.xval   2.16886    0.10752      0  20.172   <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 5552  on 4005  degrees of freedom
 Residual Deviance: 3555  on 4001  degrees of freedom

AIC: 3563  BIC: 3588  (Smaller is better. MC Std. Err. = 0.08763)
```

The p-value associated with GWESP.fixed.1.1 is 0.714, which is larger than p=0.05, suggesting that the GWESP count with a decay parameter of 1.1 is not statistically significant in predicting the network structure.

The p-value associated with GWD.fixed.2 is 0.397, which is larger than 0.05, indicating that the GWD count with a fixed degree parameter of 2 is not statistically significant in predicting the network structure.

The p-value associated with node attribute is much smaller than 0.001, indicating that the node attribute effect based on the 'xval' node attribute is highly significant in predicting the network structure.

Conclusively, GWESP and GWD counts are not statistically significant predictors of the network structure in this model. The node match effect 'xval' attribute is highly significant.

(f) Provide some comments on what you observed in Questions 4(d) and (e)

For ERGM without node information, the p-values for the GWESP and GWD is less than chosen significance level 0.05, which means the term is significant. The corresponding network structures, that is, triangles for GWESP and common neighbors for GWD, are statistically significant in that network.

However, for ERGM with Node Information, there will additionally include node information, more than GWESP and GWD. GWESP or GWD terms is no longer significant including node attributes. This suggests that network structures are important beyond the effects explained by node attributes. The nodal attribute term "xval" significantly influences the network structure instead.

Question 5:

This is a problem to understand whether the graph Laplacian is effective in generating lost node information. Consider the adjacency matrix A generated in Question 3.

(a) Apply the graph Laplacian on A and generate the values of x_i minimizing the graph Laplacian. Note that there may be isolated nodes or nodes lying in small components. For those nodes set x_i = 0.

```
> ############################ (a)
> A.lap = diag(rowSums(A)) - A
> lambda = eigen(A.lap)
> isolated_nodes = which(rowSums(A) == 0)
> x = rep(0, 100)
> lambda_x = lambda$values[abs(lambda$values) > 1e-10]
> i = which.min(abs(lambda_x))
> min_eigenvector = lambda$vectors[, i]
> non_n = setdiff(1:100, isolated_nodes)
> x[non_n] = min_eigenvector[non_n]
> x
  [1]  0.1007418546 -0.0869081316 -0.0688438966  0.0132874428 -0.0038967952 -0.0508224712 -0.0937448366
  [8] -0.0157586196 -0.0047373251  0.4619197036  0.0326468917  0.0358026387  0.0743135793  0.0029533890
 [15]  0.0811471244 -0.0610455583  0.0341614334  0.6410939531 -0.0748625198 -0.0585734330 -0.0955054704
 [22]  0.0181113985  0.0899498648 -0.0283653314 -0.0831709019  0.0274975991  0.0746782074  0.0446573508
 [29]  0.0522839118 -0.0742478148 -0.0850374329  0.0235292574 -0.0005642127  0.0291331977  0.0075329270
 [36] -0.0151908470  0.0113254514  0.0407147190 -0.0650567333 -0.0839335577  0.0045482453 -0.0700730605
 [43] -0.0124903462  0.0402647334 -0.0184907195 -0.0456729041 -0.0129766680 -0.0704095183  0.0237502481
 [50]  0.0000000000  0.0715402335 -0.0794662215 -0.0523441442 -0.0789043560  0.0477221276 -0.0805001194
 [57]  0.0216799480 -0.0720533450 -0.0804149978 -0.0033022164 -0.0863827891 -0.0476714656  0.1062029573
 [64]  0.0894450969 -0.0702022561 -0.0767694433 -0.0053454183  0.0000000000 -0.0603802241 -0.0443180950
 [71] -0.0868303074 -0.0528562773 -0.0680816805  0.0636405206 -0.1053199461 -0.0758500369  0.0554363277
 [78]  0.0287631882  0.1103113479 -0.0099617004  0.1053523229  0.0777053810  0.0000000000 -0.0088686613
 [85]  0.0000000000 -0.0805335251 -0.0065264278  0.0989190307 -0.0829269627  0.0289055353 -0.0972744972
 [92] -0.0721440973  0.0282260469 -0.0288051144 -0.0098181125 -0.0735336835 -0.0787424269 -0.0349958799
 [99]  0.0379611571  0.1536471904
```

(b) Compute the correlation between x_i and y_i. Is the correlation significantly different from zero? Apply the Fisher transformation of Tutorial 1 to compute the two-sided p-value.

```
> ############################ (b)
> corr = cor(x, y)
> fisher = 0.5 * log((1 + corr) / (1 - corr))
> n = 100 - sum(is.na(x))
> se = 1 / sqrt(n - 3)
> z = fisher / se
> p_value = 2 * (1 - pnorm(abs(z)))
> corr
[1] 0.6912985
> p_value
[1] 0
```

According to the results, the p-value is less than 0.05, which means the correlation between x_i and y_i is significantly different from zero. The correlation is 0.6912985, indicating a positive relationship between x_i and y_i.