Assignment 1 - Network Modeling

Group 5

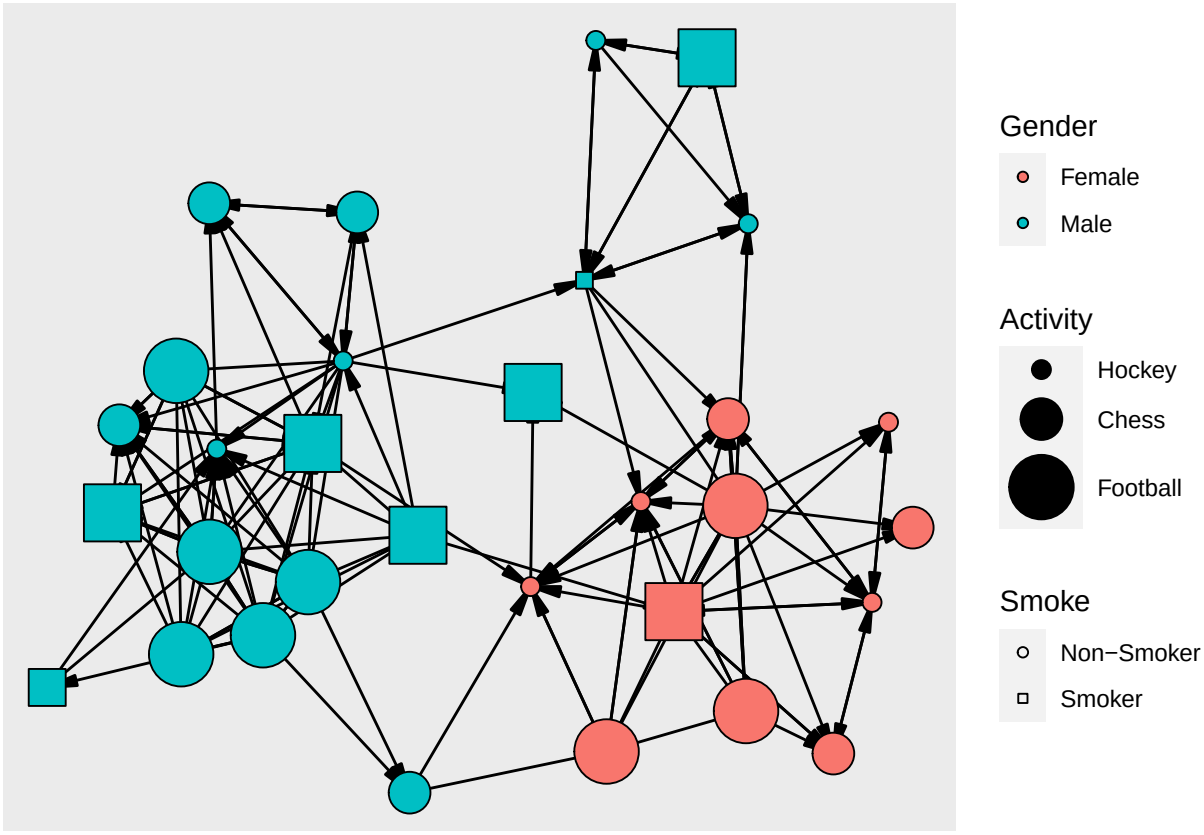
Lorenzo Benedetti (23-956-451) Francesco Freni (23-955-248)
Aristotelis Koutris (23-942-683) Ruben Oliveira Rodrigues (21-922-042)

2023-11-05

```
load('friend_net.Rda')
friend_net %v% 'smoke' <- attributes$smoke
friend_net %v% 'activity' <- attributes$activity

ggraph(friend_net) +
  geom_edge_link0(
    arrow = arrow(
      angle = 10,
      length = unit(4, "mm"),
      type = "closed"
    )
  ) +
  geom_node_point(
    aes(
      shape = as.factor(smoke),
      fill = as.factor(sex),
      size = as.factor(activity)
    ),
    colour = "black"
  ) +
  scale_fill_discrete(
    name = "Gender",
    labels = c(
      "0" = "Female",
      "1" = "Male"
    )
  ) +
  guides(fill = guide_legend(
    override.aes = list(shape = 21),
    labels = c(
      "0" = "Female",
      "1" = "Male"
    )
  )) +
  scale_shape_manual(
    name = "Smoke",
    values = c("0" = 21, "1" = 22),
    labels = c("0" = "Non-Smoker", "1" = "Smoker")
  ) +
```

```
  scale_size_manual(
    name = "Activity",
    values = c("1" = 3, "2" = 7, "3" = 11),
    labels = c("1" = "Hockey", "2" = "Chess", "3" = "Football")
  )
```

```
## Using "stress" as default layout
```



Task 1

(1)

```
set.seed(1234)
permutations <- 1000

adjMatrix <- as.matrix(friend_net)
gender <- network::get.vertex.attribute(friend_net, 'sex')
sameGender <- outer(gender, gender, "==") * 1
rownames(sameGender) <- 1:nrow(adjMatrix)
rownames(sameGender) <- 1:ncol(adjMatrix)

fit1 <- netlogit(adjMatrix, sameGender, reps = permutations, nullhyp = "qapx")
fit1$names <- c('Intercept', 'sameGender')
```

```r
# In this case qapy is equivalent to qapx, because we have only one covariate
summary(fit1)
```

```
##
## Network Logit Model
##
## Coefficients:
##             Estimate  Exp(b)      Pr(<=b) Pr(>=b) Pr(>=|b|)
## Intercept  -3.663562  0.02564103 1       1       1
## sameGender  2.676175 14.52941176 1       0       0
##
## Goodness of Fit Statistics:
##
## Null deviance: 1289.254 on 930 degrees of freedom
## Residual deviance: 675.8591 on 928 degrees of freedom
## Chi-Squared test of fit improvement:
##   613.3947 on 2 degrees of freedom, p-value 0
## AIC: 679.8591    BIC: 689.5295
## Pseudo-R^2 Measures:
##  (Dn-Dr)/(Dn-Dr+dfn): 0.3974322
##  (Dn-Dr)/Dn: 0.475775
## Contingency Table (predicted (rows) x actual (cols)):
##
##       0     1
## 0   786   144
## 1     0     0
##
##  Total Fraction Correct: 0.8451613
##  Fraction Predicted 1s Correct: NaN
##  Fraction Predicted 0s Correct: 0.8451613
##  False Negative Rate: 1
##  False Positive Rate: 0
##
## Test Diagnostics:
##
##  Null Hypothesis: qapx
##  Replications: 1000
##  Distribution Summary:
##
##         Intercept sameGender
## Min    -11.997941  -3.392107
## 1stQ   -11.997941  -0.883773
## Median -11.997941  -0.158131
## Mean   -11.997941   0.007988
## 3rdQ   -11.997941   0.749289
## Max    -11.997941   4.639983
```

The hypothesis is supported since the coefficient for `sameGender` is significantly greater than 0 with a p-value of 0. This is coherent with the plot, where we can see clusters according to the gender.

(2)
- For the first hypothesis, we have to look to the out-degree of boys. Hence, we created a matrix in which an element equals 1 if the sender is a boy, 0 otherwise.

- For the second hypothesis, we have to look to the in-degree of smokers. Hence, we created a matrix in which an element equals 1 if the receiver is a smoker, 0 otherwise.
- For the third hypothesis, we have to look for homophily with respect to the activity. Hence, we created a matrix in which an element equals 1 if the sender and the receiver take part in the same activity, 0 otherwise.

```
n <- nrow(attributes)
# Boys are more likely to send friendship nominations than girls
genderSender <- matrix(gender, n, n, byrow = FALSE)
# Smokers are more likely to receive friendship nominations than non-smokers.
smoke <- attributes$smoke
smokeReceiver <- matrix(smoke, n, n, byrow = TRUE)
# A friendship nomination is more likely between a pair of students pariticipating in
# the same activity
activity <- attributes$activity
sameActivity <- outer(activity, activity, "==") * 1

zm <- list(
  sameGender = sameGender,
  genderSender = genderSender,
  smokeReceiver = smokeReceiver,
  sameActivity = sameActivity
)
```

(3)

```
set.seed(1234)
fit2 <- netlogit(adjMatrix, zm, reps = permutations, nullhyp = "qapspp")
fit2$names <- c('Intercept', 'sameGender', 'genderSender', 'smokeReceiver',
                'sameActivity')
summary(fit2)
```

```
##
## Network Logit Model
##
## Coefficients:
##                Estimate   Exp(b)      Pr(<=b) Pr(>=b) Pr(>=|b|)
## Intercept     -3.5452552  0.02886126 0.000   1.000   0.000
## sameGender     2.9092556 18.34313856 1.000   0.000   0.000
## genderSender  -0.5834843  0.55795091 0.069   0.931   0.128
## smokeReceiver -0.3962334  0.67284964 0.111   0.889   0.218
## sameActivity   0.5542016  1.74055079 0.989   0.011   0.016
##
## Goodness of Fit Statistics:
##
## Null deviance: 1289.254 on 930 degrees of freedom
## Residual deviance: 658.3235 on 925 degrees of freedom
## Chi-Squared test of fit improvement:
##   630.9303 on 5 degrees of freedom, p-value 0
## AIC: 668.3235    BIC: 692.4994
## Pseudo-R^2 Measures:
##  (Dn-Dr)/(Dn-Dr+dfn): 0.4042014
```

```
##  (Dn-Dr)/Dn: 0.4893763
## Contingency Table (predicted (rows) x actual (cols)):
##
##       0     1
## 0   786   144
## 1     0     0
##
##  Total Fraction Correct: 0.8451613
##  Fraction Predicted 1s Correct: NaN
##  Fraction Predicted 0s Correct: 0.8451613
##  False Negative Rate: 1
##  False Positive Rate: 0
##
## Test Diagnostics:
##
##  Null Hypothesis: qapspp
##  Replications: 1000
##  Distribution Summary:
##
##         Intercept sameGender genderSender smokeReceiver sameActivity
## Min    -10.005017  -3.043011    -4.692859     -3.717527    -3.362190
## 1stQ    -5.545251  -0.798269    -1.192338     -0.981226    -0.765788
## Median  -4.375456  -0.118839     0.087359     -0.020427    -0.093415
## Mean    -4.245799   0.031071     0.010927     -0.040066     0.005256
## 3rdQ    -3.050803   0.683650     1.209218      0.797712     0.697101
## Max      1.768654   5.967086     4.562568      4.120200     5.028557
```

All the parameters, except `genderSender` and `smokeReceiver` are significantly different from 0 at a significance level $\alpha = 0.05$. Thus, we do not have evidence to reject hypothesis 1, so we do not have evidence to state that boys are more likely to send friendship nominations than girls. Also, we do not have empirical evidence that smokers are more likely to receive friendship nominations than non-smokers.

On the other hand, the other parameters are significant. In particular, the odds of having a link between students of the same gender are increased by a factor of 18.34, holding all the other variables constant. Moreover, the odds of having a link between students participating in the same extracurricular activity are 1.74 times grater than the odds of a link between students not taking part in the same activity, holding all the other variables fixed.

(4) We can test the hypothesis whether chess players are less likely to receive friendship nominations than non chess players. For this reason, we created a matrix in which an element equals 1 if the receiver plays chess, 0 otherwise.

```r
plays_chess <- ifelse(attributes$activity == 2, 1, 0)
chessReceiver <- matrix(plays_chess, n, n, byrow = TRUE)
```

(5)

```r
zm <- list(
  sameGender = sameGender,
  genderSender = genderSender,
  smokeReceiver = smokeReceiver,
  sameActivity = sameActivity,
  chessReceiver = chessReceiver
```

```
)
fit3 <- netlogit(adjMatrix, zm, reps = permutations, nullhyp = "qapspp")
fit3$names <- c('Intercept', 'sameGender', 'genderSender', 'smokeReceiver',
                'sameActivity', 'chessReceiver')
summary(fit3)
```

```
##
## Network Logit Model
##
## Coefficients:
##               Estimate   Exp(b)      Pr(<=b) Pr(>=b) Pr(>=|b|)
## Intercept     -3.4330018  0.03228987 0.000   1.000   0.000
## sameGender     2.9119419 18.39248076 1.000   0.000   0.000
## genderSender  -0.5792422  0.56032284 0.081   0.919   0.156
## smokeReceiver -0.4554945  0.63413430 0.064   0.936   0.154
## sameActivity   0.5142072  1.67231211 0.974   0.026   0.032
## chessReceiver -0.3675952  0.69239741 0.110   0.890   0.261
##
## Goodness of Fit Statistics:
##
## Null deviance: 1289.254 on 930 degrees of freedom
## Residual deviance: 655.9133 on 924 degrees of freedom
## Chi-Squared test of fit improvement:
##   633.3404 on 6 degrees of freedom, p-value 0
## AIC: 667.9133    BIC: 696.9244
## Pseudo-R^2 Measures:
##  (Dn-Dr)/(Dn-Dr+dfn): 0.40512
##  (Dn-Dr)/Dn: 0.4912458
## Contingency Table (predicted (rows) x actual (cols)):
##
##       0     1
## 0   786   144
## 1     0     0
##
##  Total Fraction Correct: 0.8451613
##  Fraction Predicted 1s Correct: NaN
##  Fraction Predicted 0s Correct: 0.8451613
##  False Negative Rate: 1
##  False Positive Rate: 0
##
## Test Diagnostics:
##
##  Null Hypothesis: qapspp
##  Replications: 1000
##  Distribution Summary:
##
##         Intercept sameGender genderSender smokeReceiver sameActivity
## Min    -10.087460  -2.756535    -6.740667     -3.309964    -3.284857
## 1stQ    -4.593659  -0.829568    -1.137973     -0.866665    -0.819793
## Median  -3.422173  -0.182732     0.074847     -0.005453    -0.044393
## Mean    -3.426445  -0.033626     0.032370      0.055447     0.016697
## 3rdQ    -2.218908   0.635964     1.360061      0.963468     0.742228
## Max      1.926569   5.748610     4.673151      3.992938     4.453664
```

```
##        chessReceiver
## Min        -3.235078
## 1stQ       -0.931616
## Median      0.013370
## Mean        0.069126
## 3rdQ        0.983088
## Max         4.152400
```

The coefficient is not significantly different from 0 (p-value 0.261), so we do not have evidence to say that chess players are less likely to be nominated as friends.

Task 2

(1)

```
#' Simulate the next step of a network in Markov chain using Metropolis-Hasting
#'
#' The function `MHstep` simulates the the Metropolis-Hastings step that defines
#' the Markov chain whose stationary distribution is the ERGM with
#' edge, mutual and nodematch statistics
#'
#' @param net an object of class `matrix`. Adjacency matrix of the network.
#' @param nodeAttr a character or numeric vector. The node attribute.
#' @param theta1 a numeric value. The value of the edge parameter of the ERGM.
#' @param theta2 a numeric value. The value of the mutual parameter of the ERGM.
#' @param theta3 a numeric value. The value of the nodematch parameter of the ERGM.
#'
#' @return next state of the Markov Chain
#'
#' @examples
#' MHstep(
#'   matrix(c(0, 1, 0, 0, 0, 0, 1, 1, 0), nrow = 3, ncol = 3),
#'   c("v", "g", "g"),
#'   -log(0.5), log(0.4), log(.8)
#' )
MHstep <- function(net, nodeAttr, theta1, theta2, theta3){

  # Number of vertices in the network
  nvertices <- nrow(net)

  # Choose randomly two vertices, prevent loops {i,i} with replace = FALSE
  tie <- sample(1:nvertices, 2, replace = FALSE)
  i <- tie[1]
  j <- tie[2]

  # Compute the change statistics
  # Number of edges
  delta_1 = 1 - 2*net[i,j]

  # Reciprocal dyads
  delta_2 = (1 - 2 * net[i,j]) * net[j,i]
  # if (net[j,i] == 1 & net[i,j] == 1) delta_2 = -1
  # if (net[i,j] == 0 & net[j,i] == 1) delta_2 = 1
```

```r
  # if (net[i,j] == 0 & net[j,i] == 0) delta_2 = 0

  # Homophily dyads
  sameAttr <- (nodeAttr[i] == nodeAttr[j]) * 1
  delta_3 = (1 - 2*net[i,j])*sameAttr

  # Compute the probability of the next state
  # according to the Metropolis-Hastings algorithm
  p <- min(1, exp(theta1 * delta_1 + theta2 * delta_2 + theta3 * delta_3))

  # Select the next state:
  ret <- rbinom(1, size=1, prob=p)
  if (ret == 1) {
    net[i,j] <- 1 - net[i,j]
  }

  # Return the next state of the chain
  return(net)
}

# Markov Chain simulation ------------------------------------------------
#' The function MarkovChain simulates the networks from the ERGM with
#' edge, mutual and nodematch statistics
#'
#' @param net an object of class `matrix`. Adjacency matrix of the network.
#' @param nodeAttr a character or numeric vector. The node attribute.
#' @param theta1 a numeric value. The value of the edge parameter of the ERGM.
#' @param theta2 a numeric value. The value of the mutual parameter of the ERGM.
#' @param theta3 a numeric value. The value of the nodematch parameter of the ERGM.
#' @param burnin an integer value.
#'   Number of steps to reach the stationary distribution.
#' @param thinning an integer value. Number of steps between simulated networks.
#' @param nNet an integer value. Number of simulated networks to return as output.
#'
#' @return a named list:
#'   - netSim: an `array` with the adjancency matrices of the simulated networks.
#'   - statSim: a `matrix` with the value of the statistic defining the ERGM.
#'
#' @examples
#' MarkovChain(
#'   matrix(c(0, 1, 0, 0, 0, 0, 1, 1, 0), nrow = 3, ncol = 3),
#'   c("v", "g", "g"),
#'   -log(0.5), log(0.4), log(.8)
#' )
MarkovChain <- function(net, nodeAttr, theta1, theta2, theta3,
                        burnin = 10000, thinning = 1000, nNet = 1000){

  # Burnin phase: repeating the steps of the chain "burnin" times
  nvertices <- nrow(net)
  burninStep <- 1 # counter for the number of burnin steps

  # Perform the burnin steps
  while (burninStep <= burnin) {
```

```
    net <- MHstep(net, nodeAttr, theta1, theta2, theta3)
    burninStep <- burninStep + 1
  }

  # After the burnin phase we draw the networks
  # The simulated networks and statistics are stored in the objects
  # netSim and statSim
  netSim <- array(0, dim = c(nvertices, nvertices, nNet))
  statSim <- matrix(0, nNet, 3)
  thinningSteps <- 0 # counter for the number of thinning steps
  netCounter <- 1 # counter for the number of simulated network

  netSim[,,netCounter] <- net
  while (netCounter < nNet) {
    thinningSteps <- 0
    while (thinningSteps < thinning) {
      net <- MHstep(net, nodeAttr, theta1, theta2, theta3)
      thinningSteps <- thinningSteps + 1
    }
    netSim[,,netCounter] <- net
    edges <- sum(net)
    mutual <- 1/2 * sum(diag(net %*% net))
    homophily <- sum(net * outer(nodeAttr, nodeAttr, '=='))
    statSim[netCounter,] <- c(edges, mutual, homophily)
    netCounter <- netCounter + 1
  }

  # Return the simulated networks and the statistics
  return(list(netSim = netSim, statSim = statSim))
}
```

(2)

(i)

```
theta1 <- -2.76; theta2 <- 0.68; theta3 <- 1.21
ret <- MarkovChain(adjMatrix, gender, theta1, theta2, theta3)
```

(ii)

```
ret$statSim %>% colMeans
```

```
## [1] 125.026  15.886  97.643
```

```
apply(ret$statSim, 2, sd)
```

```
## [1] 11.325970  3.912328  9.552214
```

```
edges <- sum(adjMatrix)
mutual <- 1/2 * sum(diag(adjMatrix %*% adjMatrix))
homophily <- sum(adjMatrix * outer(gender, gender, '=='))
c(edges, mutual, homophily)
```

```
## [1] 144  36 133
```

The observed number of mutual edges is two standard deviations away from the mean of the number of mutual edges in the simulated networks. Similar arguments can be done for the homophily term. Therefore, the given parameters are not appropriate.

(3) Considering the results from the previous point, we should increase the value of the second parameter, because we want more mutual edges in our simulated networks. Similarly, the third value should be increased as well. By increasing the second and third parameter, we noticed that we also had to decrease the first parameter. After several trials, we propose the following guesses:

```
theta1.1 <- -3.75; theta2.1 <- 2; theta3.1 <- 2
ret.1 <- MarkovChain(adjMatrix, gender, theta1.1, theta2.1, theta3.1)
ret.1$statSim %>% colMeans
```

```
## [1] 134.986  35.443 123.214
```

Task 3

(1)

```
set.seed(1234)
fit.ergm <- ergm(friend_net ~ edges + nodematch('sex'))
```

```
## Starting maximum pseudolikelihood estimation (MPLE):

## Obtaining the responsible dyads.

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.

## Evaluating log-likelihood at the estimate.
```

```
summary(fit.ergm)
```

```
## Call:
## ergm(formula = friend_net ~ edges + nodematch("sex"))
##
## Maximum Likelihood Results:
##
##               Estimate Std. Error MCMC % z value Pr(>|z|)
## edges          -3.6636     0.3053      0 -11.998   <1e-04 ***
## nodematch.sex   2.6762     0.3218      0   8.316   <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 1289.3  on 930  degrees of freedom
##  Residual Deviance:  675.9  on 928  degrees of freedom
##
## AIC: 679.9  BIC: 689.5  (Smaller is better. MC Std. Err. = 0)
```

Adding a tie between two nodes of the same gender increases both the edges and homophily statistics by 1. As a result the log-odds of a tie is equal to $\theta_1 + \theta_2$.

```
theta1 <- coef(fit.ergm)[1]
names(theta1) <- NULL
theta2 <- coef(fit.ergm)[2]
names(theta2) <- NULL
odds <- exp(theta1 + theta2)
prob <- odds/(1 + odds)
prob
```

```
## [1] 0.2714286
```

Therefore, the probability of observing a tie between students of the same gender is 27%. As the estimate of the second parameter is significantly greater than zero at a significance level $\alpha = 0.05$, we can state that having the same gender increases the probability of being friends. This is coherent with the following result, which is the probability of having a tie given that the nodes do not have the same gender:

```
exp(theta1)/(1+exp(theta1))
```

```
## [1] 0.025
```

(2) For (i) we should add the number of mutual edges; for (ii) the number of transitive two-path, considering decreasing weight of triangles to avoid near-degeneracy; for (iii) we should add a term encoding for the out-degree distribution; for (iv) we should add a term encoding for the in-degree distribution. Also for the last two terms we consider adding a decay parameter.

(3)

```
set.seed(1234)
fit.ergm2 <- ergm(friend_net ~ edges + mutual + nodematch('sex') +
                    gwesp(decay = 0.3, fixed = TRUE) +
                    gwodegree(decay = 0.3, fixed = TRUE) +
                    gwidegree(decay = 0.3, fixed = TRUE))
```

```
## Starting maximum pseudolikelihood estimation (MPLE):

## Obtaining the responsible dyads.

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.

## Starting Monte Carlo maximum likelihood estimation (MCMLE):

## Iteration 1 of at most 60:
```

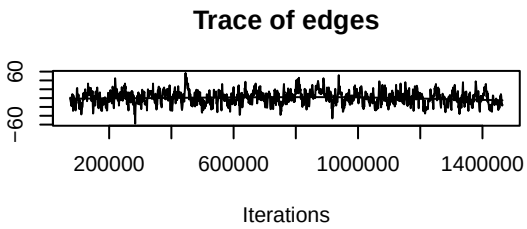
```
## Warning: 'glpk' selected as the solver, but package 'Rglpk' is not available;
## falling back to 'lpSolveAPI'. This should be fine unless the sample size and/or
## the number of parameters is very big.

## Optimizing with step length 0.2345.

## The log-likelihood improved by 2.7872.

## Estimating equations are not within tolerance region.

## Iteration 2 of at most 60:

## Optimizing with step length 0.2816.

## The log-likelihood improved by 1.9308.

## Estimating equations are not within tolerance region.

## Iteration 3 of at most 60:

## Optimizing with step length 0.6554.

## The log-likelihood improved by 2.9834.

## Estimating equations are not within tolerance region.

## Iteration 4 of at most 60:

## Optimizing with step length 1.0000.

## The log-likelihood improved by 0.4673.

## Estimating equations are not within tolerance region.

## Iteration 5 of at most 60:

## Optimizing with step length 1.0000.

## The log-likelihood improved by 0.0763.

## Convergence test p-value: 0.7547. Not converged with 99% confidence; increasing sample size.
## Iteration 6 of at most 60:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0423.
## Convergence test p-value: 0.5286. Not converged with 99% confidence; increasing sample size.
## Iteration 7 of at most 60:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0523.
## Convergence test p-value: 0.4219. Not converged with 99% confidence; increasing sample size.
```

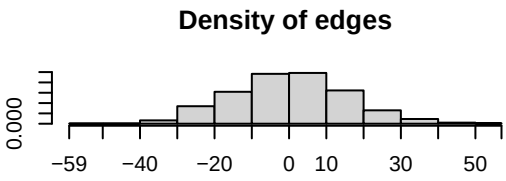
```
## Iteration 8 of at most 60:
## Optimizing with step length 1.0000.
## The log-likelihood improved by 0.0115.
## Convergence test p-value: 0.0090. Converged with 99% confidence.
## Finished MCMLE.
## Evaluating log-likelihood at the estimate. Fitting the dyad-independent submodel...
## Bridging 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.
```

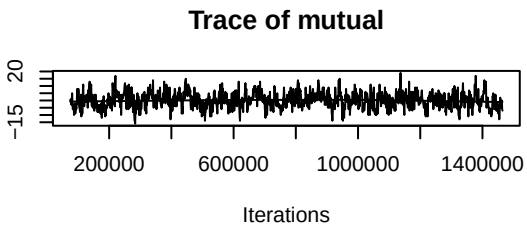
```
summary(fit.ergm2)
```

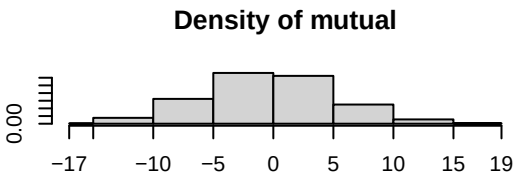
```
## Call:
## ergm(formula = friend_net ~ edges + mutual + nodematch("sex") +
##     gwesp(decay = 0.3, fixed = TRUE) + gwodegree(decay = 0.3,
##     fixed = TRUE) + gwidegree(decay = 0.3, fixed = TRUE))
##
## Monte Carlo Maximum Likelihood Results:
##
##                     Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                -5.7500     0.4158      0 -13.827  < 1e-04 ***
## mutual                0.7978     0.3409      0   2.340  0.01928 *
## nodematch.sex         0.9561     0.2123      0   4.503  < 1e-04 ***
## gwesp.OTP.fixed.0.3   2.0966     0.3138      0   6.682  < 1e-04 ***
## gwodeg.fixed.0.3      1.3403     0.7482      0   1.791  0.07325 .
## gwideg.fixed.0.3      2.7510     0.9793      0   2.809  0.00497 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 1289.3  on 930  degrees of freedom
##  Residual Deviance:  568.8  on 924  degrees of freedom
##
## AIC: 580.8  BIC: 609.8  (Smaller is better. MC Std. Err. = 0.6526)
```

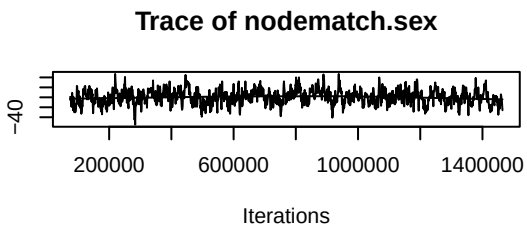
```
mcmc.diagnostics(fit.ergm2)
```

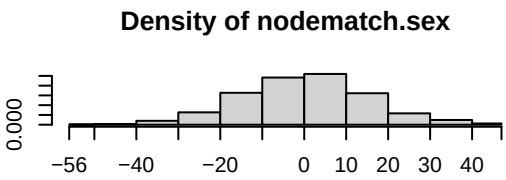

Trace of edges
−60
60
200000
600000
1000000
1400000
Iterations

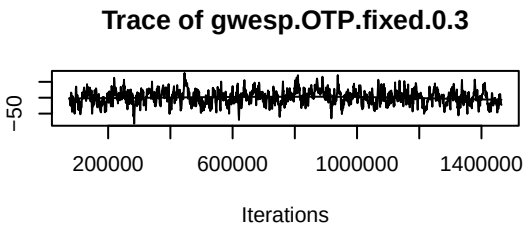

Density of edges
0.000
−59
−40
−20
0
10
30
50

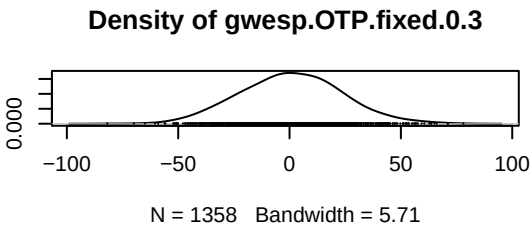

Trace of mutual
−15
20
200000
600000
1000000
1400000
Iterations

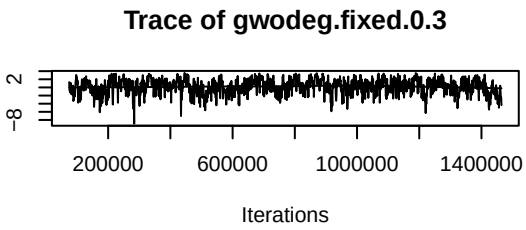

Density of mutual
0.00
−17
−10
−5
0
5
10
15
19

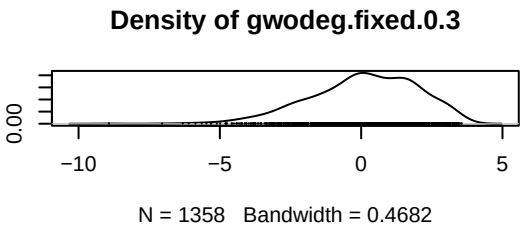

Trace of nodematch.sex
−40
200000
600000
1000000
1400000
Iterations

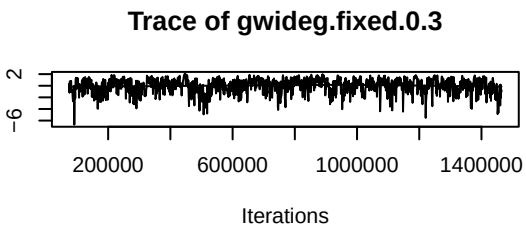

Density of nodematch.sex
0.000
−56
−40
−20
0
10
20
30
40

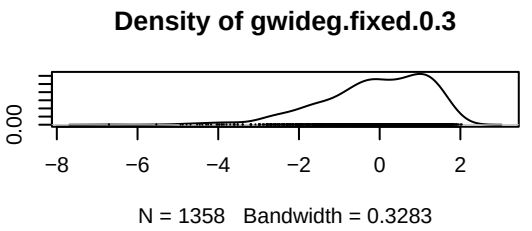










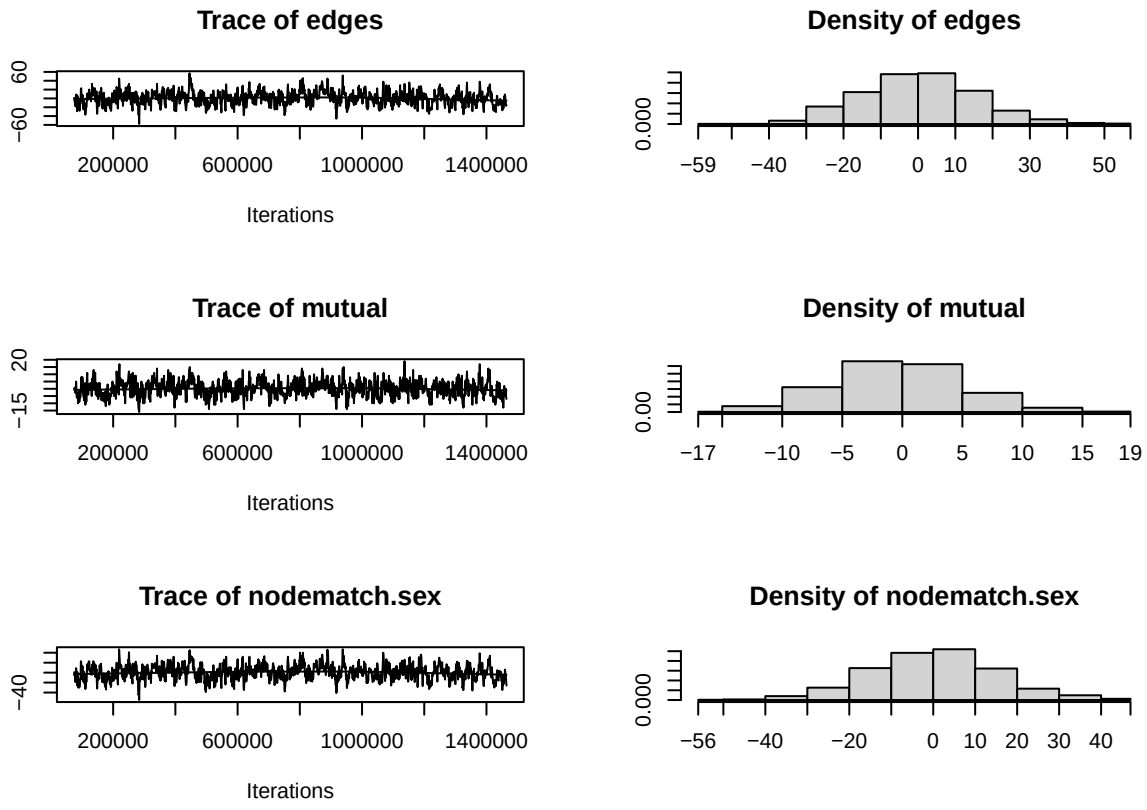


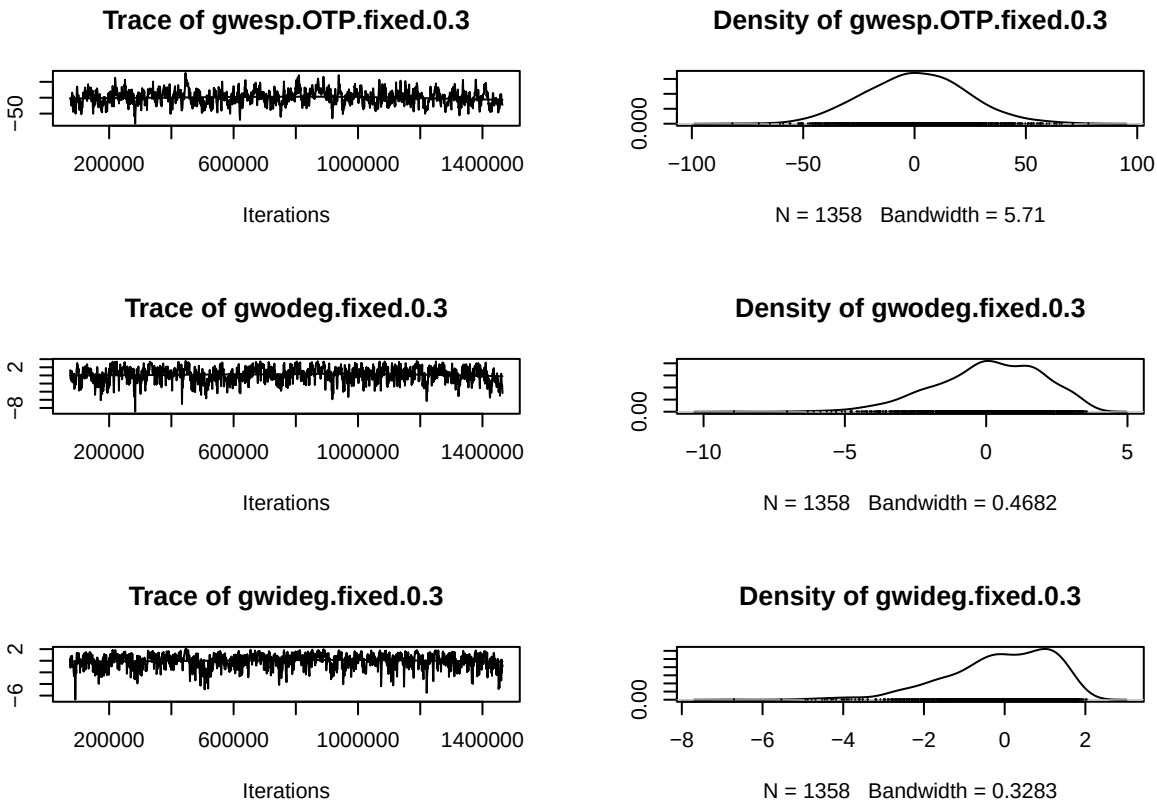

```
## Sample statistics summary:
##
## Iterations = 74752:1464320
## Thinning interval = 1024
## Number of chains = 1
## Sample size per chain = 1358
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                       Mean     SD Naive SE Time-series SE
## edges                0.5722 15.413  0.41826        1.04585
## mutual               0.1156  5.410  0.14681        0.35531
## nodematch.sex        0.4278 15.011  0.40735        1.05773
## gwesp.OTP.fixed.0.3  0.7502 22.798  0.61867        1.51617
## gwodeg.fixed.0.3     0.1005  1.870  0.05073        0.12094
## gwideg.fixed.0.3    -0.1070  1.311  0.03557        0.08838
##
## 2. Quantiles for each variable:
##
##                        2.5%      25%     50%     75%  97.5%
## edges               -28.000 -10.0000 1.00000 11.0000 32.000
## mutual              -10.000  -4.0000 0.00000  4.0000 11.000
## nodematch.sex       -29.000  -9.7500 1.00000 10.0000 31.000
## gwesp.OTP.fixed.0.3 -42.944 -14.7105 0.39675 16.0766 45.391
## gwodeg.fixed.0.3     -3.860  -1.1102 0.21153  1.5452  3.141
```

```
## gwideg.fixed.0.3     -2.976  -0.8591 0.04767  0.9504  1.712
##
##
## Are sample statistics significantly different from observed?
##                edges    mutual nodematch.sex gwesp.OTP.fixed.0.3
## diff.      0.5721649 0.1156112     0.4278351           0.7502097
## test stat. 0.5470789 0.3253813     0.4044853           0.4948055
## P-val.     0.5843245 0.7448925     0.6858559           0.6207375
##            gwodeg.fixed.0.3 gwideg.fixed.0.3       (Omni)
## diff.             0.1005234       -0.1070053           NA
## test stat.        0.8311574       -1.2108053 2.304336e+01
## P-val.            0.4058847        0.2259700 9.805544e-04
##
## Sample statistics cross-correlations:
##                         edges    mutual nodematch.sex gwesp.OTP.fixed.0.3
## edges                1.0000000 0.7721419     0.9322090           0.9835612
## mutual               0.7721419 1.0000000     0.7927820           0.8118061
## nodematch.sex        0.9322090 0.7927820     1.0000000           0.9415679
## gwesp.OTP.fixed.0.3 0.9835612 0.8118061     0.9415679           1.0000000
## gwodeg.fixed.0.3     0.5978590 0.5092832     0.6092322           0.5520518
## gwideg.fixed.0.3     0.5671055 0.4796464     0.5694077           0.5363111
##                      gwodeg.fixed.0.3 gwideg.fixed.0.3
## edges                       0.5978590        0.5671055
## mutual                      0.5092832        0.4796464
## nodematch.sex               0.6092322        0.5694077
## gwesp.OTP.fixed.0.3         0.5520518        0.5363111
## gwodeg.fixed.0.3            1.0000000        0.5096399
## gwideg.fixed.0.3            0.5096399        1.0000000
##
## Sample statistics auto-correlation:
## Chain 1
##              edges    mutual nodematch.sex gwesp.OTP.fixed.0.3 gwodeg.fixed.0.3
## Lag 0    1.0000000 1.0000000     1.0000000           1.0000000        1.0000000
## Lag 1024 0.6317400 0.5153723     0.6492759           0.6221173        0.4495211
## Lag 2048 0.4566748 0.3613800     0.4859091           0.4495307        0.3634033
## Lag 3072 0.3142705 0.3010112     0.3686476           0.3192684        0.2883004
## Lag 4096 0.2426119 0.2712890     0.2924168           0.2525124        0.2681037
## Lag 5120 0.1760196 0.1902084     0.2257737           0.1893464        0.2079114
##          gwideg.fixed.0.3
## Lag 0           1.0000000
## Lag 1024        0.4015610
## Lag 2048        0.3048517
## Lag 3072        0.2141052
## Lag 4096        0.2150268
## Lag 5120        0.1513951
##
## Sample statistics burn-in diagnostic (Geweke):
## Chain 1
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##               edges              mutual       nodematch.sex gwesp.OTP.fixed.0.3
##           -1.550292           -1.339893           -2.060814           -1.856889
```

```
##    gwodeg.fixed.0.3    gwideg.fixed.0.3
##           -2.029321           -2.186138
##
## Individual P-values (lower = worse):
##               edges              mutual       nodematch.sex gwesp.OTP.fixed.0.3
##          0.12107137          0.18028010          0.03932075          0.06332693
##    gwodeg.fixed.0.3    gwideg.fixed.0.3
##          0.04242562          0.02880554
## Joint P-value (lower = worse):  0.02312545
##
## Note: MCMC diagnostics shown here are from the last round of
##   simulation, prior to computation of final parameter estimates.
##   Because the final estimates are refinements of those used for this
##   simulation run, these diagnostics may understate model performance.
##   To directly assess the performance of the final model on in-model
##   statistics, please use the GOF command: gof(ergmFitObject,
##   GOF=~model).
```

```
plot(fit.ergm2$sample)
```





Adding the geometrically weighted parameters was necessary to reach convergence. Moreover, considering the results obtained by the function `mcmc.diagnostics`, we can conclude that the algorithm has reached convergence. The MCMC is mixing well, since the difference between the observed and simulated statistics fluctuates around 0. Additionally, the probability distributions of the errors are centered around 0.

(4) We use the `gof` function and consider criteria and auxiliary statistics (e.g. minimum geodesic distance) which were not taken into consideration while fitting the model.

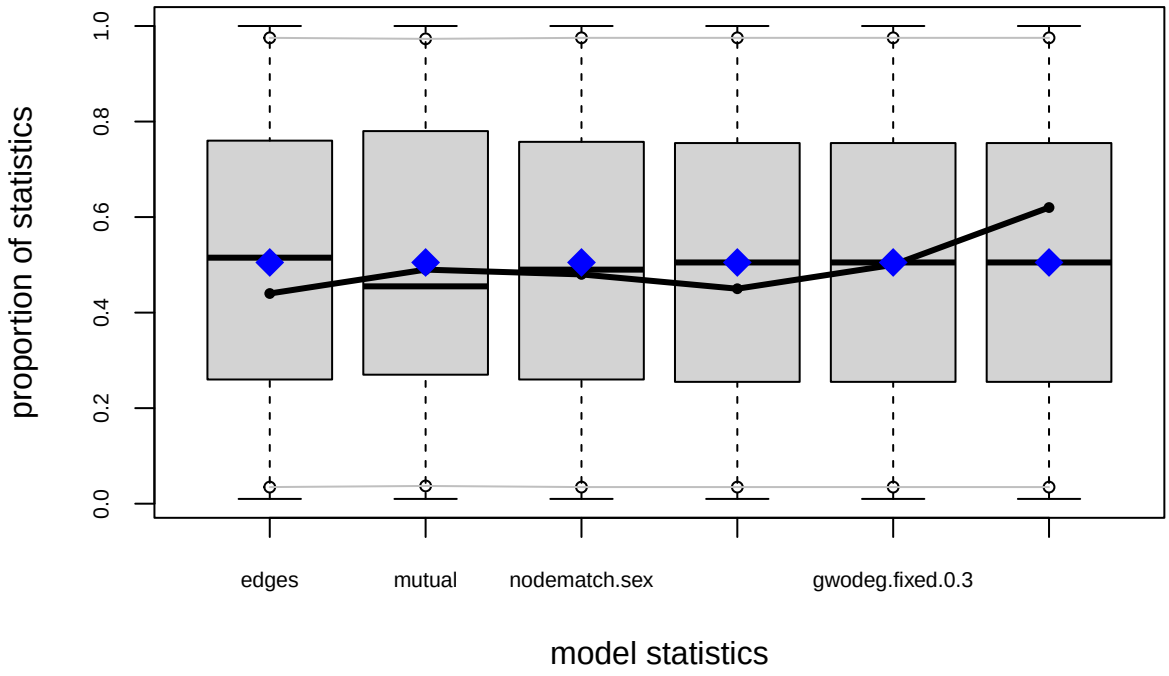
```
gof.fit.ergm2 <- gof(fit.ergm2)
gof.fit.ergm2
```

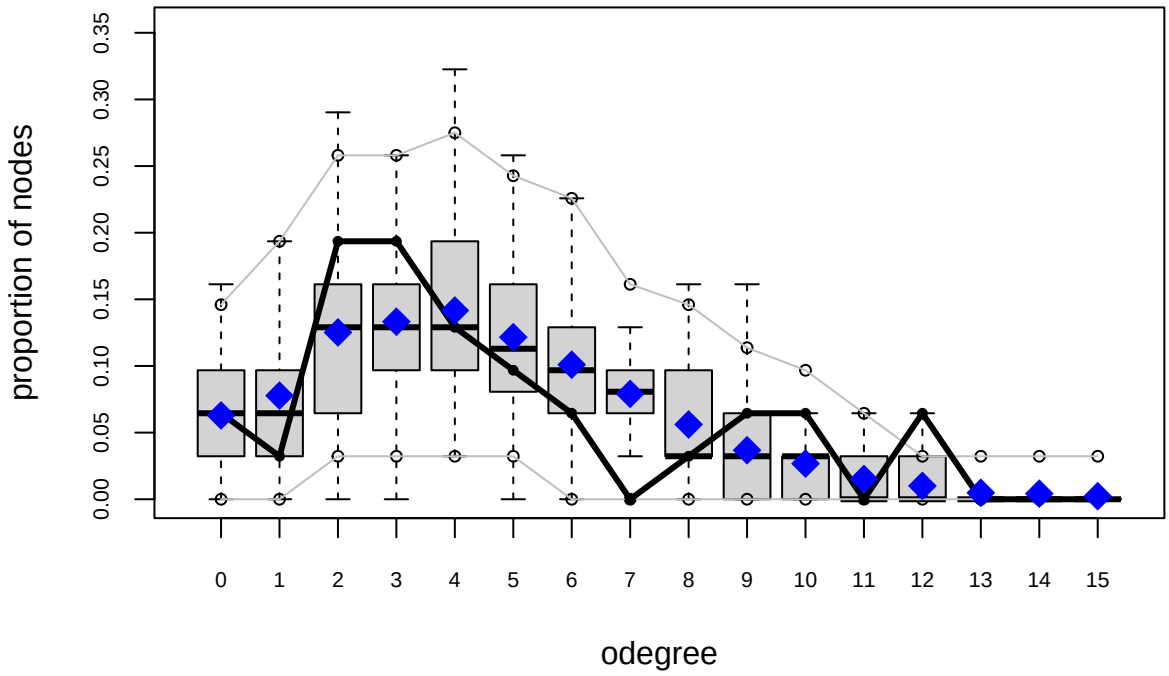
```
##
## Goodness-of-fit for in-degree
##
##           obs min mean max MC p-value
## idegree0    1   0 0.50   2       0.78
## idegree1    1   0 2.43   8       0.72
## idegree2    3   0 4.55  13       0.68
## idegree3    6   1 4.99  10       0.74
## idegree4    6   1 4.34   8       0.44
## idegree5    4   1 4.39  10       1.00
## idegree6    2   0 3.15   7       0.88
## idegree7    5   0 2.23   5       0.06
## idegree8    1   0 1.80   6       1.00
## idegree9    0   0 1.09   5       0.72
## idegree10   2   0 0.56   3       0.20
```

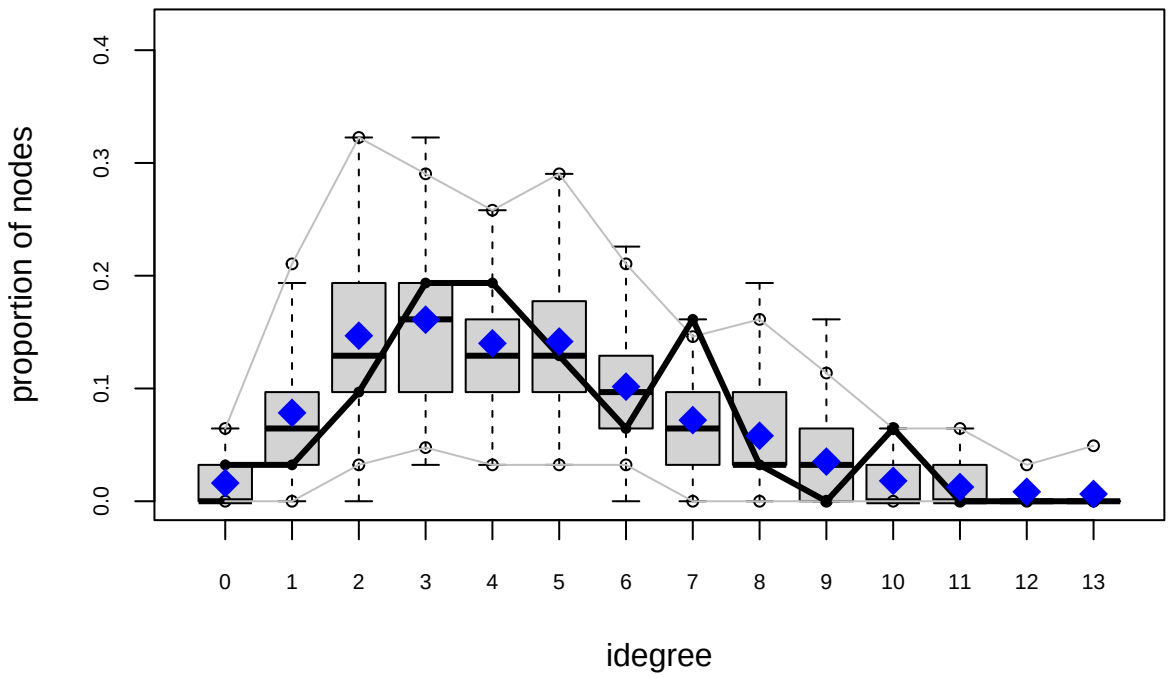
```
## idegree11   0   0 0.39   2       1.00
## idegree12   0   0 0.26   2       1.00
## idegree13   0   0 0.20   2       1.00
## idegree14   0   0 0.03   1       1.00
## idegree15   0   0 0.07   1       1.00
## idegree16   0   0 0.02   1       1.00
##
## Goodness-of-fit for out-degree
##
##           obs min mean max MC p-value
## odegree0    2   0 1.95   5       1.00
## odegree1    1   0 2.41   7       0.64
## odegree2    6   0 3.88   9       0.42
## odegree3    6   0 4.13   9       0.42
## odegree4    4   1 4.39  11       1.00
## odegree5    3   0 3.77  10       1.00
## odegree6    2   0 3.13   9       0.74
## odegree7    0   0 2.45   6       0.16
## odegree8    1   0 1.74   5       1.00
## odegree9    2   0 1.14   5       0.64
## odegree10   2   0 0.83   5       0.40
## odegree11   0   0 0.47   2       1.00
## odegree12   2   0 0.31   2       0.02
## odegree13   0   0 0.15   2       1.00
## odegree14   0   0 0.13   1       1.00
## odegree15   0   0 0.07   1       1.00
## odegree16   0   0 0.02   1       1.00
## odegree17   0   0 0.02   1       1.00
## odegree18   0   0 0.01   1       1.00
##
## Goodness-of-fit for edgewise shared partner
##
##          obs min  mean max MC p-value
## esp.OTP0  11   1  9.90  19       0.88
## esp.OTP1  44  33 46.98  63       0.72
## esp.OTP2  35  28 45.50  67       0.32
## esp.OTP3  16   5 24.29  46       0.26
## esp.OTP4  13   0 10.17  26       0.70
## esp.OTP5  12   0  3.61  12       0.02
## esp.OTP6  11   0  1.08   6       0.00
## esp.OTP7   2   0  0.29   2       0.06
## esp.OTP8   0   0  0.09   1       1.00
## esp.OTP9   0   0  0.02   1       1.00
##
## Goodness-of-fit for minimum geodesic distance
##
##     obs min   mean max MC p-value
## 1   144  98 141.93 189       0.88
## 2   173 162 278.48 403       0.04
## 3   138  58 188.21 282       0.36
## 4    76   0  98.55 181       0.64
## 5    70   0  44.03 127       0.44
## 6    31   0  15.03  71       0.32
## 7     1   0   4.11  51       0.82
```

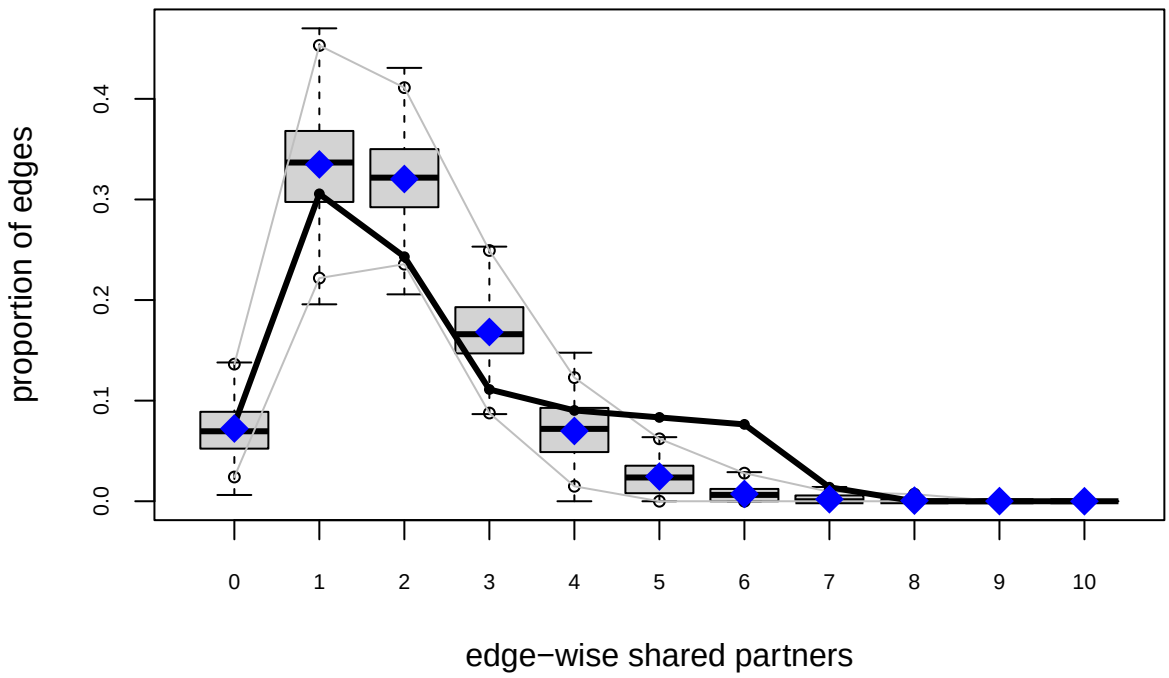
```
## 8     0   0   0.87  14       1.00
## 9     0   0   0.10   4       1.00
## 10    0   0   0.01   1       1.00
## Inf 297   0 158.68 541       0.34
##
## Goodness-of-fit for model statistics
##
##                            obs      min      mean       max MC p-value
## edges                144.00000 98.00000 141.93000 189.00000       0.88
## mutual                36.00000 22.00000  35.36000  50.00000       0.98
## nodematch.sex        133.00000 91.00000 130.91000 172.00000       0.96
## gwesp.OTP.fixed.0.3 160.48486 91.66642 157.02070 229.00040       0.90
## gwodeg.fixed.0.3     38.08101 33.47701  37.88706  41.31330       1.00
## gwideg.fixed.0.3     39.68862 36.83012  39.75573  41.52157       0.76
```

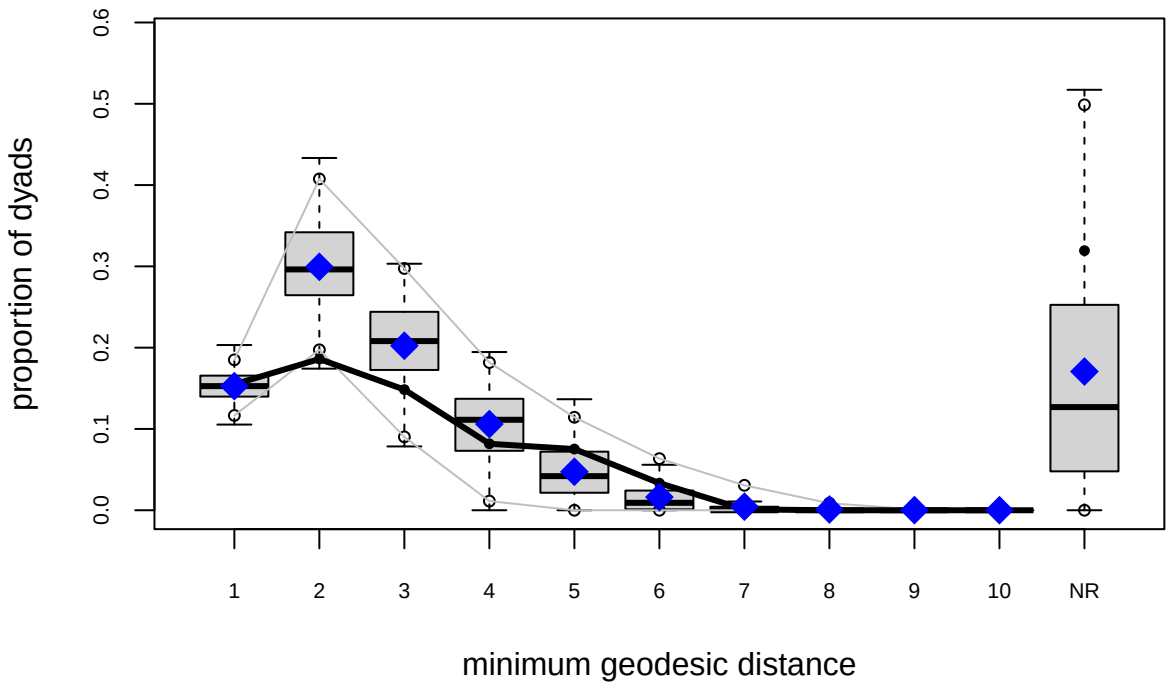
```
plot(gof.fit.ergm2)
```




proportion of nodes
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
odegree


proportion of nodes
0.0
0.1
0.2
0.3
0.4
0
1
2
3
4
5
6
7
8
9
10
11
12
13
idegree


proportion of edges
0.0
0.1
0.2
0.3
0.4
0
1
2
3
4
5
6
7
8
9
10
edge−wise shared partners



Overall, the MCMC is good at modelling the auxiliary statistics. Considering the out-degree, there is only one value outside the 95% confidence interval, and the same holds true for the in-degree. As for the edge-wise shared partners statistics, only two values are outside the confidence interval, while for the minimum geodesic distance all the values are inside the confidence interval.

```
observed <- summary(friend_net ~ edges + mutual + nodematch('sex') +
                    gwesp(decay = 0.3, fixed = TRUE) +
                    gwodegree(decay = 0.3, fixed = TRUE) +
                    gwidegree(decay = 0.3, fixed = TRUE))

newcoef <- coef(fit.ergm2)
simNetsStat <- simulate(friend_net ~ edges + mutual + nodematch('sex') +
                    gwesp(decay = 0.3, fixed = TRUE) +
                    gwodegree(decay = 0.3, fixed = TRUE) +
                    gwidegree(decay = 0.3, fixed = TRUE),
                    nsim = 1000,
                    coef = newcoef, output = "stats"
                    )
expected <- apply(simNetsStat, 2, mean)
observed
```

```
##               edges              mutual       nodematch.sex gwesp.OTP.fixed.0.3
##           144.00000            36.00000           133.00000           160.48486
##    gwodeg.fixed.0.3    gwideg.fixed.0.3
##            38.08101            39.68862
```

```
expected
```

```
##                edges               mutual       nodematch.sex gwesp.OTP.fixed.0.3
##            141.62500             35.08400            130.87400            157.06517
##    gwodeg.fixed.0.3    gwideg.fixed.0.3
##             37.73712             39.61593
```

We notice that the simulated statistics are similar to the observed ones.

(5) Considering the summary, the estimated parameters are significant with $\alpha = 0.05$, except the fifth. The negative value for the edge parameter is coherent to what we expect for social networks (the network is sparse).

- The mutual parameter is positive and significant indicating that reciprocal ties are more likely. This supports the hypothesis (i), providing evidence for the presence of reciprocity in the data.
- The geometrically weighted edgewise shared partners parameter is positive and therefore, there is evidence for transitivity and hypothesis (ii) is also supported by the data.
- We do not have evidence to state that a tie is less likely between students when the sender has a higher out-degree. Therefore, we reject hypothesis (iii).
- The geometrically weighted in-degree is positive and significant, so there is evidence to say that a tie is more likely between students when the receiver has a higher in-degree and we do not reject hypothesis (iv).

Task 4

(1)

```
set.seed(1234)
fit.ergm3 <- ergm(friend_net ~ edges + mutual + nodematch('sex') +
                    gwesp(decay = 0.3, fixed = TRUE) +
                    gwodegree(decay = 0.3, fixed = TRUE) +
                    gwidegree(decay = 0.3, fixed = TRUE) +
                    nodeofactor('sex') +
                    nodeifactor('smoke') +
                    nodematch('activity'),
                  control=control.ergm(main.method='Stochastic-Approximation'))
```

```
## Starting maximum pseudolikelihood estimation (MPLE):

## Obtaining the responsible dyads.

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.

## Stochastic approximation algorithm with theta_0 equal to:
```

```
##               edges              mutual       nodematch.sex gwesp.OTP.fixed.0.3
##         -5.04522555          0.31757648          1.54968768          1.68256821
##    gwodeg.fixed.0.3    gwideg.fixed.0.3   nodeofactor.sex.1 nodeifactor.smoke.1
##         -0.47128729         -0.27774402          0.03282804         -0.30099506
##  nodematch.activity
##          0.60107024
## Starting burnin of 16384 steps
## Phase 1: 200 steps (interval = 1024)

## Stochastic Approximation estimate:

##               edges              mutual       nodematch.sex gwesp.OTP.fixed.0.3
##          -5.2464634           0.8030678           1.1527472           1.8033115
##    gwodeg.fixed.0.3    gwideg.fixed.0.3   nodeofactor.sex.1 nodeifactor.smoke.1
##           0.6973061           1.8618446          -0.3272455          -0.2372490
##  nodematch.activity
##           0.3954076

## Phase 3:  1000 iterations (interval=1024)
## 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(fit.ergm3)
```

```
## Call:
## ergm(formula = friend_net ~ edges + mutual + nodematch("sex") +
##     gwesp(decay = 0.3, fixed = TRUE) + gwodegree(decay = 0.3,
##     fixed = TRUE) + gwidegree(decay = 0.3, fixed = TRUE) + nodeofactor("sex") +
##     nodeifactor("smoke") + nodematch("activity"), control = control.ergm(main.method = "Stochastic-Ap
##
## Stochastic Approximation Maximum Likelihood Results:
##
##                     Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                -5.4888     0.4151      0 -13.224   <1e-04 ***
## mutual                0.7352     0.3349      0   2.195   0.0282 *
## nodematch.sex         1.0641     0.2478      0   4.294   <1e-04 ***
## gwesp.OTP.fixed.0.3   1.9998     0.3072      0   6.509   <1e-04 ***
## gwodeg.fixed.0.3      0.9279     0.7996      0   1.161   0.2458
## gwideg.fixed.0.3      2.0853     1.1068      0   1.884   0.0596 .
## nodeofactor.sex.1    -0.2871     0.1573      0  -1.825   0.0680 .
## nodeifactor.smoke.1  -0.2430     0.1917      0  -1.267   0.2050
## nodematch.activity    0.3678     0.1580      0   2.327   0.0200 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 1289.3  on 930  degrees of freedom
##  Residual Deviance:  556.8  on 921  degrees of freedom
```

```
##
## AIC: 574.8  BIC: 618.3  (Smaller is better. MC Std. Err. = 0.496)
```

```
set.seed(1234)
fit.ergm3 <- ergm(friend_net ~ edges + nodematch('sex') +
                    nodeofactor('sex') +
                    nodeifactor('smoke') +
                    nodematch('activity'))
```

```
## Starting maximum pseudolikelihood estimation (MPLE):

## Obtaining the responsible dyads.

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.

## Evaluating log-likelihood at the estimate.
```

```
summary(fit.ergm3)
```

```
## Call:
## ergm(formula = friend_net ~ edges + nodematch("sex") + nodeofactor("sex") +
##     nodeifactor("smoke") + nodematch("activity"))
##
## Maximum Likelihood Results:
##
##                     Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                -3.5453     0.3296      0 -10.756  < 1e-04 ***
## nodematch.sex         2.9093     0.3325      0   8.750  < 1e-04 ***
## nodeofactor.sex.1    -0.5835     0.2225      0  -2.622  0.00873 **
## nodeifactor.smoke.1  -0.3962     0.2334      0  -1.697  0.08964 .
## nodematch.activity    0.5542     0.2035      0   2.724  0.00645 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 1289.3  on 930  degrees of freedom
##  Residual Deviance:  658.3  on 925  degrees of freedom
##
## AIC: 668.3  BIC: 692.5  (Smaller is better. MC Std. Err. = 0)
```

Without the structural terms, the `ergm` function uses MPLE, so the estimates are the same as in Task 1, but the p-values are different. In this case, we reject the hypothesis that smokers are more likely to receive friendship nominations than non-smokers given that $\alpha = 0.05$. On the other hand, we have evidence of homophily based on the type of activity. Finally, although the nodeofactor.sex parameter is statistically significant, its value is negative which indicates that boys are less likely, rather than more likely, to send friendship nominations. As a result, we reject the corresponding hypothesis.

On the other hand, if we add the structural terms, we get different estimates. In particular, the out-degree and in-degree distribution of the whole network can adequately explain the observed network, so

`nodeofactor.sex.1` and `nodeifactor.smoke.1` are found to be not significant. Specifically, the impact of the out-degree statistic is significant but the same cannot be said about the out-degree statistic of boys. Similarly, the in-degree statistic is significant, while the in-degree statistic of smokers is not.

(2) We can test whether football players reciprocate friendship nominations they get from chess players. The mathematical formula would be the following:

$$\sum_{i,j} x_{ij} \cdot (1 - x_{ji}) \cdot \mathbb{I}(v_i == 2) \cdot \mathbb{I}(v_j == 3),$$

where $\mathbb{I}$ is the indicator function.