Assignment1

Mirco Büchel 22-933-097 TASK1
Kailin Liu 21-951-470 TASK3
Qinghao Guan 21-750-260 TASK3
Zhengxu Li 22-944-482 TASK2

Task1

(1) Give the mathematical formula for each effect. You can use the RSiena manual to look for the formulas.

Out-degree: $s_{1i}(x) = \sum_j x_{ij}$

where $x_{ij} = 1$ indicates presence of a tie from i to j while $x_{ij} = 0$ indicates absence of this tie

Reciprocity: $s_{2i}(x) = \sum_j x_{ij}\, x_{ji}$

Transitive reciprocated triplets: $s_{3i}(x) = \sum_{j,h} x_{ij}\, x_{ji} x_{ih} x_{hj}$

Same Covariate: $s_{4i}(x, v) = \sum_j x_{ij}\, I(v_i = v_j)$

where the indicator function $I(v_i = v_j)$ is 1 if the condition $\{v_i = v_j\}$ is satisfied, and 0 if it is not;

(2) Given the current state of the network, (...) what is the probability that in the next mini-step:

In the following, the tables show the values of every effect (columns) after toggling the tie to the other actors (rows). A dot represents no change in the network.
These are then multiplied with the given beta-values, to calculate the objective function.
Lastly, we follow the formula $P(i \to j;\, x, \beta) = \frac{exp(f(i, x^{\pm ij}, \beta))}{\sum_k exp(f(i, x^{\pm ik}, \beta))}$ to get the probability.

i. actor c adds a tie to b ?

	s_1	s_2	s_3	s_4
a	0	0	0	0
b	2	1	1	0
•	1	0	0	0
d	2	0	0	1

$\Rightarrow f(c, x^{+ca}, \beta) = 0$

$f(c, x^{+cb}, \beta) = -3 + 2 + 1 = 0$

$f(c, x, \beta) = -1,5$

$f(c, x^{+cd}, \beta) = -3 + 1,5 = -1,5$

$\Rightarrow \frac{e^0}{e^0 + e^0 + e^{-1,5} + e^{-1,5}} = 0,408$

ii. actor b adds a tie to a ?

	s_1	s_2	s_3	s_4
a	2	1	1	1
•	1	0	0	0
c	0	0	0	0
d	2	1	0	0

$\Rightarrow f(b, x^{+ba}, \beta) = -3 + 2 + 1 + 1,5 = 1,5$

$f(b, x, \beta) = -1,5$

$f(b, x^{+bc}, \beta) = 0$

$f(b, x^{+bd}, \beta) = -3 + 2 = -1$

$\Rightarrow \frac{e^{1,5}}{e^{1,5} + e^{-1,5} + e^0 + e^{-1}} = 0,738$

iii. actor a deletes the tie to b ?

	s_1	s_2	s_3	s_4
•	2	1	0	1
b	1	1	0	0
c	3	2	1	1
d	1	0	0	1

$\Rightarrow f(a, x, \beta) = -3 + 2 + 1,5 = 0,5$

$f(a, x^{+ab}, \beta) = -1,5 + 2 = 0,5$

$f(a, x^{+ac}, \beta) = -4,5 + 4 + 1 + 1,5 = 2$

$f(a, x^{+ad}, \beta) = -1,5 + 1,5 = 0$

$\Rightarrow \frac{e^{0,5}}{e^{0,5} + e^{0,5} + e^2 + e^0} = 0,141$

iv. actor d does not change anything?

	s_1	s_2	s_3	s_4
a	1	0	0	0
b	1	1	0	0
c	3	1	1	1
•	2	1	0	0

$\Rightarrow f(a, x^{+da}, \beta) = -1,5$

$f(a, x^{+db}, \beta) = -1,5 + 2 = 0,5$

$f(a, x^{+dc}, \beta) = -4,5 + 2 + 1 + 1,5 = 0$

$f(a, x, \beta) = -3 + 2 = -1$

$\Rightarrow \frac{e^{-1}}{e^{-1,5} + e^{0,5} + e^0 + e^{-1}} = 0,113$

Task 2: Simulations from SAOM

The file simSAOMs.R contains the code to simulate the network evolution between two observations from a SAOM with an evaluation function specified by outdegree, reciprocity and transitive triplets effects statistics. It also includes the code to produce violin plots for the triad census counts.

(1) Implement the missing code so that the function simulation can be used to simulate the network evolution. Document the code. The algorithm is described in the file Simulating from SAOM available in the Lecture notes and additional material section on Moodle. Unconditional simulation is used.

Hint: a useful function for the implementation is sample

Tip: If you want to implement an efficient code for the simulation, you can use change statistics (How much would the statistic change if the tie is toggled?). In this way, creating the network with the toggled tie is unnecessary for computing the effect statistics. For transitive triplets, it is useful to consider that a tie from a sender can play one of two roles in this network structure.

```r
simulation <- function(n, x1, lambda, beta1, beta2, beta3) {
  t <- 0 # time
  x <- x1
  while (t < 1) {
    dt <- rexp(1, n * lambda)
    # --- MISSING ---
    i <- sample(1:n, 1)  # Select an actor at random
    #iterate through all the j's to calculate the evaluation function if the change was made by i,j
    vector_p_ij <-  vector("numeric", length = n) #an vector to store the change probability
    for (j in 1:n){
      if (j==i){change_stat <- c(0,0,0)}
      else{
        sign_change <- -2 * (x[i,j]) + 1 #map value 0 to 1, map 1 to -1,
        num_two_path <- x %*% x
        num_two_path_i_j <- num_two_path[i,j] #number of two path from i to j
        num_common_pred <- sum(x[i,]*x[j,]) #number of predecessor of i and j
        num_direct_succ <- sum(x[,i]*x[,j]) #number of successor  of i and j
        change_stat <- c(sign_change,
                         sign_change * x[j,i] ,
                         sign_change * (num_two_path_i_j + num_common_pred + num_direct_succ))
      }
      ev_fct = exp(sum(c(beta1,beta2,beta3)*change_stat)) #calculate the probability of every possible change
      vector_p_ij[j] <- ev_fct #store it in the vector of change
    }
  vector_p_ij_norm <- vector_p_ij/sum(vector_p_ij)#normalize the probability to have sum 1
  sample <- rmultinom(n=1, size=1, vector_p_ij_norm) #draw j from multinomial distribution using the calculate probability
  index_j <- which(sample == 1) #get the index of the actor to which we make the change
  if (i != index_j){
    x[i,j] <- 1- x[i,j] # if j is not i itself, we toggle the edge
  }
  else{
    x <- x # if j coincides with i, we make the network unchanged.
  }
  t <- t + dt
  }
  return(x)
}
```

Here are several note-worthy things to comment on. We describe them by the step of the algorithm.

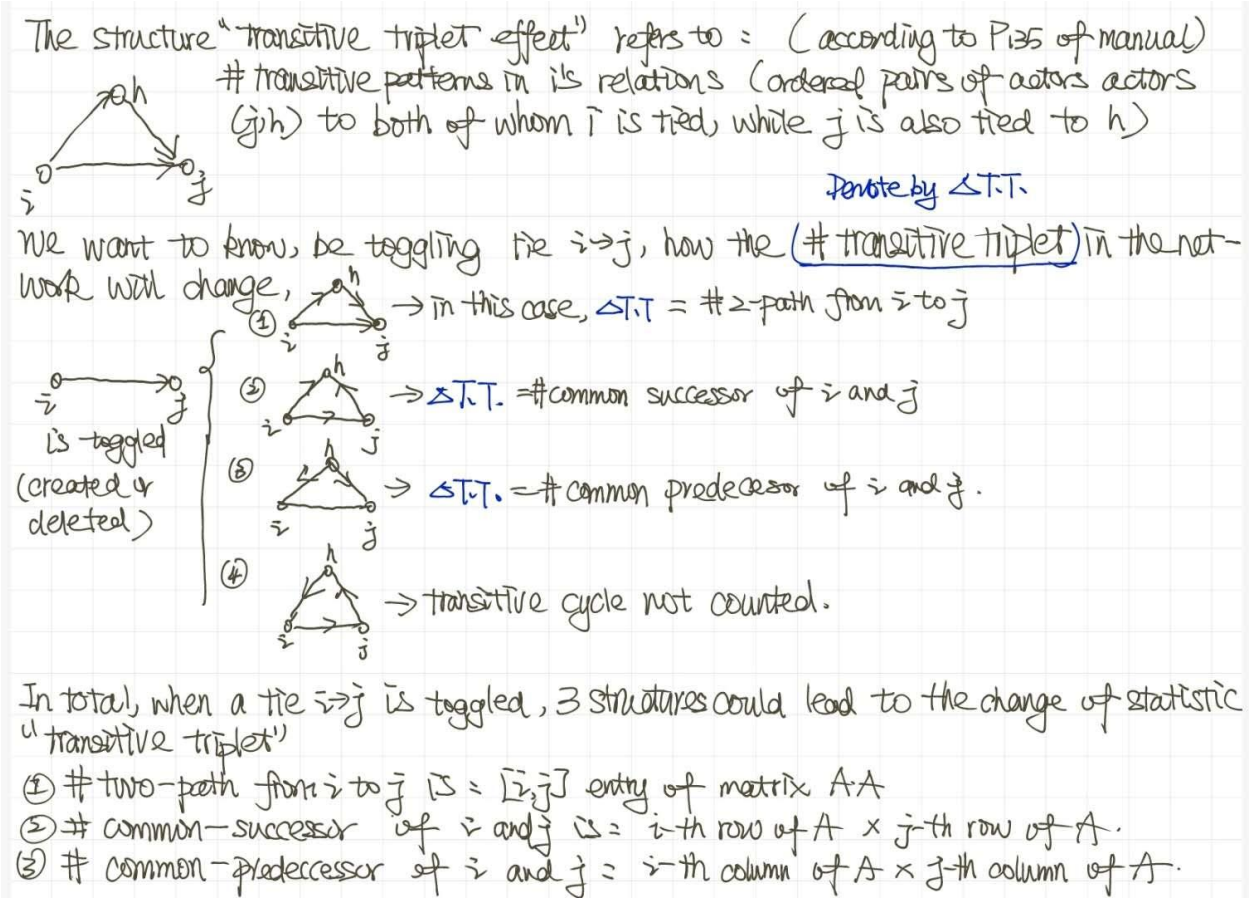
First, we sample a number from 1 to n, to decide who will be the next actor i.

Then we create a vector “vector_p_ij” to store the value of the change of evaluation function if the tie (i,j) was toggled. We iterate through all the possible j’s(including the i which refers to not making change, in this case, the change of statistic is zero vector, otherwise we enter the else loop to analyze what happens)

An interesting trick we have applied is to map the (i,j) entry of adjacency matrix x to variable “sign_change”. Specifically, if there was a tie from i to j, i.e. (i,j) = 1 in matrix x, then toggle tie(i,j) means delete it, hence we map (i,j) = 1 to a value -1, which means that statistics (like outdegree) will decrease. On the opposite, if there was no tie from i to j, i.e. (i,j) = 0 in matrix x, then toggle tie(i,j) means adding such edge, hence we map(i,j) = 0 to value 1, which means that we will increase some statistics. Such map of value(1,0) to (-1,1) was made by 2*x[i,j] + 1.

Then we analyse how the three statistics will change:

1. Outdegree: It is just the map we talked about, since if the (i,j)was present, it would be deleted, hence this statistic decrease by 1. If (i,j) was not present, it will be added, hence the statistic will increase by 1. Equivalent to taking variable “sign_change”
2. Reciprocity: Here the change depends on the presence of tie (j,i): if (j,i) was present before, adding (i,j) will increase reciprocity by 1, deleting (i,j) will decrease reciprocity by 1. If (j,i) was not present before, then no matter we add or delete tie (i,j), the reciprocity will not change. If we summarize those 4 cases, it could be resumed by the expression “sign_change * x[j,i]”
3. Transitive triplet: For this, we need some drawings to better illustrate the idea, the cases are discussed as follows:



The change of statistic for transitivity triplet could be summarized by “sign_change * (num_two_path_i_j + num_common_pred + num_direct_succ)” as what we draw in the picture, is the contribution of 3 possible structures invoked by the toggling of (i,j)

After calculating the change statistics, we just calculate the exponential function of linear combination of coefficient(betas) and change of statistics to obtain the gain of evaluation functions. We store them in the vector and then normalize them to have probabilities that sum up to 1. Then we take a sample from this probability vector based on multinomial distribution, change the tie (i,j), and continue the loop until it reaches 1.

(2) Consider the two adjacency matrices in the files net1.csv and net2.csv. They are observations of two networks collected on a set of 22 actors at time t1 and t2, respectively. Estimate the parameters of the SAOM with outdegree, reciprocity and transitive tiplets statistics using the function siena07.

We explain step by step what we have done, first we import the necessary libraries

```
> library(RSiena)
> library(sna)
```

We read the two waves

```
> net1 <- as.matrix(read.csv("net1.csv", header = FALSE))
> net2 <- as.matrix(read.csv("net2.csv", header = FALSE))
```

Then we create an object to include the two observations

```
> nets <- sienaDependent( array(c(net1, net2), dim = c(22, 22, 2)))
> mydata <- sienaDataCreate(nets)
> print01Report(mydata, modelname = "nets")
```

Here is the screenshot of the report printed after creating the object

```
@1
Initial data description.
=========================


@2
Change in networks:
-------------------

For the following statistics, missing values (if any) are not counted.

Network density indicators:
observation time             1      2
density                  0.167  0.171
average degree           3.500  3.591
number of ties              77     79
missing fraction         0.000  0.000

The average degree is 3.545


Tie changes between subsequent observations:
 periods        0 =>  0   0 =>  1   1 =>  0   1 =>  1   Distance Jaccard   Missing
  1 ==>   2       355        30        28        49        58     0.458         0 (0%)

Directed dyad Counts:
 observation    total    mutual    asymm.     null
     1.           462       46        62       354
     2.           462       30        98       334

Standard values for initial parameter values
--------------------------------------------

basic rate parameter nets              5.5214
outdegree (density)                            -0.7702

Initialisation of project <<nets>> executed succesfully.
```

Where we can see that the Jaccard index is greater than 0.3(0.458), hence it is representative of applying SAOM model in our dataset.

After than, we include the effect of transitive triplets(since the first two were already included by default) and display if we have all three required statistics(outdegree, reciprocity, transitive triplets) together with the rate parameter we build our model.

```
> myeff <- getEffects(mydata)
> myeff <- includeEffects(myeff, transTrip)
  effectName          include fix   test  initialValue parm
1 transitive triplets TRUE    FALSE FALSE          0   0
> myeff
  effectName               include fix   test  initialValue parm
1 basic rate parameter nets TRUE    FALSE FALSE    5.52138   0
2 outdegree (density)       TRUE    FALSE FALSE   -0.77022   0
3 reciprocity               TRUE    FALSE FALSE    0.00000   0
4 transitive triplets       TRUE    FALSE FALSE    0.00000   0
```

Then we create an algorithm object to specify some hyperparameter and then call the siena07() function to estimate the model.

```
> myAlgorithm <- sienaAlgorithmCreate(
+  projname = "mynets",
+  nsub = 4, n3 = 1000, seed = 2023
+ )
If you use this algorithm object, siena07 will create/use an output file
mynets.txt .
> model2_2 <- siena07(
+  myAlgorithm,
+  data = mydata, effects = myeff,
+  returnDeps = TRUE,
+  useCluster = TRUE, nbrNodes = 3
+ )
>
> model2_2
Estimates, standard errors and convergence t-ratios

                              Estimate   Standard   Convergence
                                           Error      t-ratio

Rate parameters:
  0       Rate parameter       4.0967  ( 0.6725   )

Other parameters:
  1. eval outdegree (density) -1.1067  ( 0.1992   )   -0.0443
  2. eval reciprocity          0.4750  ( 0.3094   )   -0.0455
  3. eval transitive triplets  0.0763  ( 0.0920   )   -0.0319

Overall maximum convergence ratio:    0.0507


Total of 1946 iteration steps.
```

(3) Conditioning on the first observation, generate 1,000 simulations of the network evolution using the function simulation developed in (1) and setting the parameters with the results of the model estimated in (2). Compute the triad census counts for each simulated network. Save the results in an R object1 , named triadCensus, in which rows are the index of a simulated network and columns are the type of triads.

Hint: use the function triad.census() from the sna package to compute the triad census counts.

num_simulations <- 1000

```r
# ---MISSING---

num_simulations <- 1000
# Initialize matrix for storing triad census results
num_triad_types <- length(colnames(triad.census(net1)))
triadCensus <- matrix(nrow = num_simulations, ncol = num_triad_types)
colnames(triadCensus) <- colnames(triad.census(net1))
# Run simulations
n_nodes <- 22
for (i in 1:num_simulations) {
  # Simulate network evolution
  print(i)
  netT2 <- simulation(n_nodes, net1, 4.097, -1.1067, 0.4750, 0.0763) # Use parameters of the previous model
  triad_census_result <- triad.census(netT2) # get the triad census of the simulation
  triadCensus[i, ] <- triad_census_result # Store the result in the matrix
}

```

We set the parameter using the parameters of the previous model and run 1000 iterations, and store the “triadCensus”

(4) Use the simulated values of the triad census counts to evaluate the model’s goodness of fit. The second part of the code was written to this aim, complete the missing pieces of code to produce the violin plots. Additionally, write the code to compute the Mahalanobis distance and the p-value used in RSiena to assess the fit of the model with respect to the triad census auxiliary statistic. Remember to drop statistics with variance of 0 for the plot and Mahalanobis distance computation, report which statistics suffer this issue. The code should compute the following quantities:

i. Standardize the simulated network statistics, i.e., centered and scaled values of each type of triad given in the triadCensus object. Named the resulting object as triadCensusStd. (The centered and scaled values are computed as $xstd = (x - \bar{x})/\sigma x$ with $\bar{x}$ the average and σx the standard deviation of the simulated distribution).

```r
# Task 2.4 ----------------------------------------------------------------
## i. standardized the simulated network stats. ----
##   Name the resulting object as triadCensusStd

# ---MISSING---

column_mean <- as.vector(apply(triadCensus, MARGIN = 2, mean))
column_std <- as.vector(apply(triadCensus, MARGIN = 2, sd))

triadCensus_centered <- sweep(triadCensus, MARGIN = 2, STATS = column_mean, FUN = "-")
triadCensusStd <- sweep(triadCensus_centered, MARGIN = 2, STATS = column_std, FUN = "/")
triadCensusStd

```

We first use function apply() to calculate the mean and standard deviation of every column(each type of triad census), which will be passed later to the function sweep() to recenter and standardize the triadCensus.

ii. the variance-covariance matrix of the standarized simulated network statistics $P^{\hat{}}$ and its generalized inverse. Hint: useful functions are cov() and MASS::ginv()

iii. standardize the observed values of the triad census counts in the second observation with ¯x and σx as in i.

```
174 ## ii. variance-covariance matrix and its generalized inverse.        ----
175
176 # ---MISSING---
177 var_cov <- cov(triadCensusStd)
178 gen_inv <- MASS::ginv(var_cov)
```

This step is straightforward, just plug in the functions.

iv. compute the Mahalanobis distance for each simulated and the observed network using the standardized values (computed on i. and iii.). (The Mahalanobis distance is computed as x T stdPˆ−1xstd)

```
206 mhd <- function(auxStats, invCov) {
207   t(auxStats) %*% invCov %*% auxStats
208 }
209
210 # ---MISSING---
211 #MHD of the simulations, return a vector of 1000 values
212 MHD_sim <- apply(triadCensusStd, MARGIN = 1, function(row) mhd(auxStats = row, invCov = gen_inv))
213 MHD_obs <- mhd(auxStats = triadCensus_net2_std, invCov = gen_inv)
214
```

For the simulated networks, we again use function apply() to calculate the Mahalanobis distance for each simulated network, and for the observed network we could directly call the function mhd().

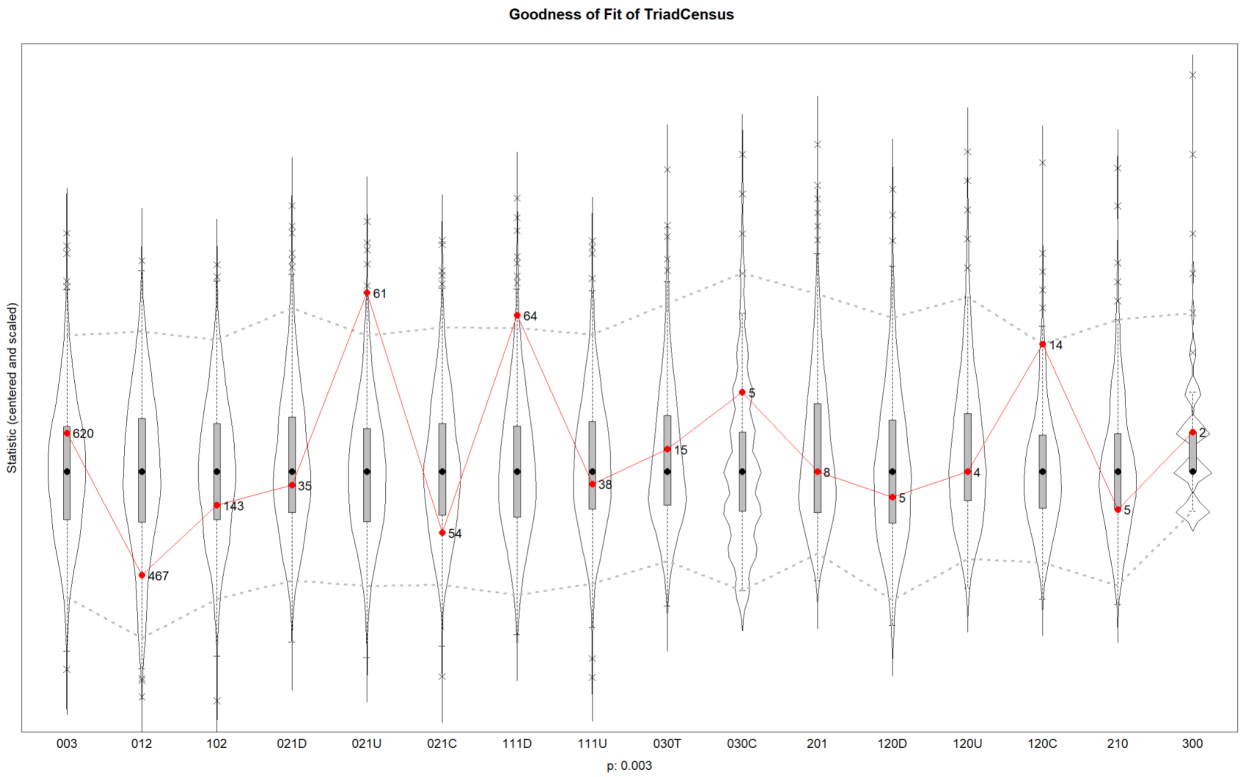
v. compute the percentage of simulated networks with Mahalanobis distance equal or greater than the observed network Mahalanobis distance.

```
216 ## iii. Monte-Carlo p-value computation                                ----
217 # Compute the proportion of simulated networks where the distance is
218 # equal or greater than the distance in the observed network.
219
220 # ---MISSING---
221 prop <- (sum(MHD_sim >= MHD_obs[1,1])) / length(MHD_sim)
222 prop
```

```
> prop <- (sum(MHD_sim >= MHD_obs[1,1])) / length(MHD_sim)
> prop
[1] 0
```

Interestingly, using the parameter (obtained by siena07()) and previously implemented simulation()functions. All 1000 simulations result in higher Mahalanobis distance than the observed network. We exhaustively checked our model and re-runed lots of times and all we got were zero proportions. This means that the model didn't have a good fit based on the triad census auxiliary statistics. If we use directly the Riesa() model to evaluate goodness of fit and plot the triad census, as follows:

```
# Triad census
gof1.tc <- sienaGOF(model2_2, verbose = TRUE,
                    varName = "nets", TriadCensus)

plot(gof1.tc, center = TRUE, scale = TRUE)
```

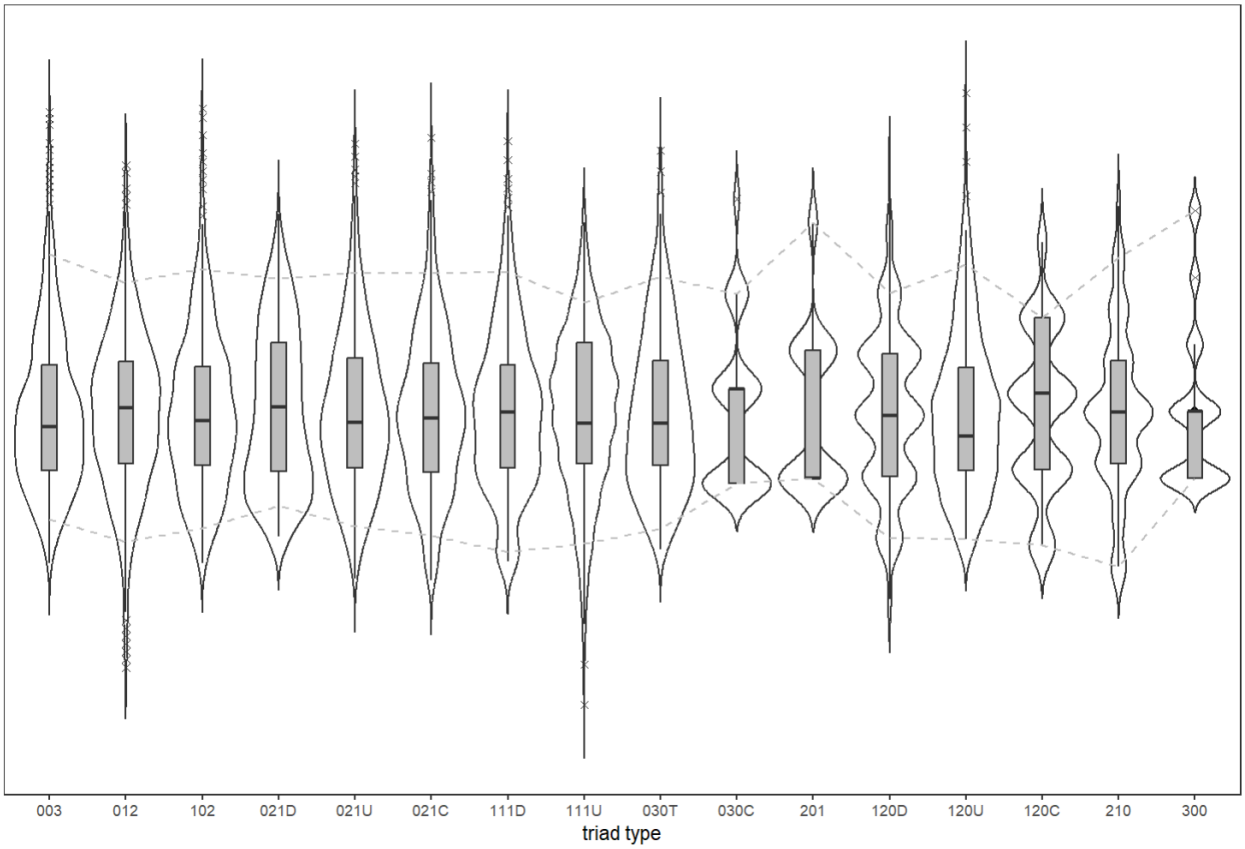


As the p-value shows, there are only 3 simulations out of 1000 that has higher Mahalanobis distance than the observed network. We deduce that the net1 and net2 are not steadily modeled by the SAOM in terms of triad census auxiliary statistics, hence both our simulation and RSiena showed poor goodness of fit.

Run the complete code to obtain the violin plots and the test on the Mahalanobis distance. Would you think that the model has a good fit based on the triad census auxiliary statistics and the p-value compute in (4)? Justify your answer.

Hint: a useful function to apply the same function to the rows or columns of a data frame(array) is apply

```
235
236 triadCensusDf <- data.frame(triadCensusStd) |>
237   select(where(~ var(.) > 0)) |>  # Drop statistics with zero variance
238   pivot_longer(
239     everything(),
240     names_to = "triad", names_pattern = "^X(.+)$",
241     values_to = "nnodes"
242   )
243
```



Indeed, like what we commented previously, there are no simulation that has higher MHD than the observed network and we observed that in the boxplots that no statistics of the observed network falled in the range of the simulations.

It could be caused by several reasons: 1) Is likely that our implementation of the simulation function was not 100% correct, but we looked through that lots of times but still can't locate the possible bug. 2) Due to the definition of "transitive triplets", in fact in the manual says that some version of RSiena has an error overcounting the "transitive triplets". In our solution, we do count it since we think that we need to calculate all the possible transitive triplets of the network produced by the change of tie (i,j) instead of just focusing on the ego node i and calculating only its relation. Imagine that we the tie (i,j) is added which creates more transitive triplets where i is the intermediate node, while it does not contribute to the statistic with respect to node i, but in the perspective of the whole network, it should get counted, so in our final solution we keep the "structure case 3" in the drawing, However, we also tried to delete that count and rerun the whole model, it still got p value 0, so the cause of poor goodness of fit is not likely by this.

> (there was an error here until version 3.313, which amounted to combining the transitive triplets and transitive mediated triplets effects;)

3)another possible explanation is like what we mentioned before, due to the inherent network structure, it does not fit well why evaluate using triad census.

Task 3: Estimation and interpretation of SAOMs

(1) Let us start by considering the friendship network as the only dependent variable

(1.1) Compute the Jaccard index to evaluate if the data contains enough information to investigate the evolution of the friendship network. Comment on the results.

The Jaccard index, also known as the Jaccard similarity coefficient, is a statistic used for gauging the similarity and diversity of sample sets. In network modeling, the Jaccard index is always used to assess the stability or change in the network over time. It is calculated as the size of the intersection divided by the size of the union of the sample sets.
Typically, a very high Jaccard index might indicate that the network is relatively stable over time and a very low Jaccard index could suggest significant changes in the network, which could be of interest for dynamic network analysis.

We used two approaches to calculate the jaccard index. In the first way, we calculate the intersection and union together within a function.

```
####  Approach 1 for calculation of the Jaccard index
# Define a function to calculate Jaccard index
jaccard_index <- function(net1, net2) {
  intersection <- sum(net1 & net2)
  union <- sum(net1 | net2)
  return(intersection / union)
}

# Calculate Jaccard index between each pair of networks
jaccard_1_2 <- jaccard_index(net1, net2)
jaccard_2_3 <- jaccard_index(net2, net3)

# Output the results
print(paste("Jaccard index between net1 and net2:", jaccard_1_2))
print(paste("Jaccard index between net2 and net3:", jaccard_2_3))
```

In the second way, we calculate the N11, N10 and N01 separately.

```r
####  Approach 2 for calculation of the Jaccard index
# Function to calculate the Jaccard index according to the provided definitions
jaccard_index <- function(net1, net2) {
  # N11: Number of ties present at both time points
  N11 <- sum(net1 & net2)
  # N10: Number of ties present at the first but absent at the second time point
  N10 <- sum(net1 & !net2)
  # N01: Number of ties absent at the first but present at the second time point
  N01 <- sum(!net1 & net2)
  # Jaccard index formula
  jaccard <- N11 / (N11 + N01 + N10)
  return(jaccard)
}

# Calculate Jaccard index between each pair of networks
jaccard_1_2 <- jaccard_index(net1, net2)
jaccard_2_3 <- jaccard_index(net2, net3)

# Output the results
print(paste("Jaccard index between net1 and net2:", jaccard_1_2))
print(paste("Jaccard index between net2 and net3:", jaccard_2_3))
```

The threshold of 0.3 is often used in practice to decide if longitudinal network data are informative enough for SAOM. For our dataset, the Jaccard index between net1 and net2 is 0.303649635036496 while the Jaccard index between net2 and net3 is 0.350668647845468.
The Jaccard index between net1 and net2 is approximately 0.304, which is just above the 0.3 threshold. This suggests that there is some stability in the network between these time points, but also enough change to potentially model the network dynamics. The Jaccard index between net2 and net3 is approximately 0.351, which is comfortably above the threshold, indicating that there's a moderate level of change that could be informative for dynamic modeling.

(1.2) Specify a reasonable model to test the following hypotheses on the friendship evolution:
i. Students tend to be friends with popular pupils
ii. Students tend to be friends with pupils with similar alcohol consumption to their own
iii. Students tend to be friends with students that live in the same neighborhood (living nearby).

```r
# Create dependent variables for a Siena model
genderCovar <- coCovar(attributes$gender)
ageCovar <- coCovar(attributes$age)
alcoholCovar <- varCovar(alcohol)

# Create a dyadic covariate object 'logdistance'
logdistanceCovar <- coDyadCovar(logdistance)

# Create the dependent variable object 'friendshipNet'
friendshipNet <- sienaDependent(array(c(net1, net2, net3), dim = c(129, 129, 3)))

# Create the RSiena data object
mySienaData <- sienaDataCreate(friendshipNet, genderCovar, ageCovar, alcoholCovar, logdistanceCovar)

#print01Report(mySienaData, modelname = "mySienaData")

# Model specification
myeff <- getEffects(mySienaData)
```

We create covariate objects for gender, age and alcohol. We also create dyadic covariate objects, such as `logdistanceCovar`. We then combine three adjacency matrices into a single array. The sienaDependent function creates a dependent variable for the Siena model, which in this case is the evolving friendship network. Then, we create a Siena data object, which is the main input for the RSiena analysis. It includes the friendship network as well as the covariates (gender, age, alcohol consumption, and geographical distance) that might affect the network dynamics. With the method of getEffects(), we retrieve the default set of effects for the specified Siena data object.
Here is the effects information.

	effectName	include	fix	test	initialValue	parm
1	constant friendshipNet rate (period 1)	TRUE	FALSE	FALSE	7.45424	0
2	constant friendshipNet rate (period 2)	TRUE	FALSE	FALSE	6.82927	0
3	outdegree (density)	TRUE	FALSE	FALSE	-1.61299	0
4	reciprocity	TRUE	FALSE	FALSE	0.00000	0

Then we tested Hypothesis 1, 2 and 3. Here are the codes below.

```r
myAlgorithm <- sienaAlgorithmCreate(
  projname = "friends_res",
  nsub = 4, n3 = 3000, seed = 1908
)
# H1: Students tend to be friends with popular pupils
effects_h1 <- includeEffects(myeff, inPop, name = "friendshipNet")
effects_h1

model1 <- siena07(
  myAlgorithm,
  data = mySienaData, effects = effects_h1,
  returnDeps = TRUE,
  useCluster = TRUE, nbrNodes = 4, batch = FALSE
)
```

```r
#H2: Students tend to be friends with pupils with similar alcohol consumption to their own
effects_h2 <- includeEffects(myeff, altX,
                             name = "friendshipNet",
                             interaction1 = "alcoholCovar")
effects_h2

model2 <- siena07(
  myAlgorithm,
  data = mySienaData, effects = effects_h2,
  returnDeps = TRUE,
  useCluster = TRUE, nbrNodes = 4, batch = FALSE
)

model2
```

```r
# H3: Students tend to be friends with students that live in the same neighborhood (living nearby)
effects_h3 <- includeEffects(myeff, altX, name = "friendshipNet",  interaction1 = "logdistanceCovar")
effects_h3

model3 <- siena07(
  myAlgorithm,
  data = mySienaData, effects = effects_h3,
  returnDeps = TRUE,
  useCluster = TRUE, nbrNodes = 4, batch = FALSE
)

model3
```

(1.3) Estimate the model, check its convergence and fit, and comment on its parameters.

We set up the estimation algorithm with specific parameters and then run the model estimation using the siena07 function.

Here is the output of our model for three hypotheses.

```
> model1
Estimates, standard errors and convergence t-ratios

                                   Estimate   Standard   Convergence
                                              Error      t-ratio

Rate parameters:
  0.1      Rate parameter period 1  8.5941   ( 0.7245   )
  0.2      Rate parameter period 2  7.2716   ( 0.5903   )

Other parameters:
  1.  eval outdegree (density)     -2.7118   ( 0.0803   )   0.0427
  2.  eval reciprocity              2.7319   ( 0.0832   )   0.0558
  3.  eval indegree - popularity    0.0627   ( 0.0142   )   0.0375

Overall maximum convergence ratio:    0.0562


Total of 3758 iteration steps.
```

In model 1, the outdegree (density) is -2.7118, and the overall maximum convergence ratio is 0.0562. Due to the negative outdegree, people do not tend to be friends with popular students.

```
> model2
Estimates, standard errors and convergence t-ratios

                                    Estimate    Standard    Convergence
                                                  Error       t-ratio

Rate parameters:
  0.1      Rate parameter period 1   8.5709   ( 0.6958   )
  0.2      Rate parameter period 2   7.2583   ( 0.5850   )

Other parameters:
  1.  eval outdegree (density)      -2.4142   ( 0.0377   )   -0.0027
  2.  eval reciprocity               2.7101   ( 0.0843   )    0.0136
  3.  eval alcoholCovar alter       -0.0177   ( 0.0254   )   -0.0291

Overall maximum convergence ratio:    0.0386


Total of 3578 iteration steps.
```

In model 2, the outdegree (density) is -2.4142, and the overall maximum convergence ratio is 0.0386. Due to the negative outdegree, students do not tend to be friends with pupils with similar alcohol consumption to their own.

```
> model3
Estimates, standard errors and convergence t-ratios

                                  Estimate    Standard    Convergence
                                                Error       t-ratio

Rate parameters:
  0.1      Rate parameter period 1   8.6025   ( 0.6901   )
  0.2      Rate parameter period 2   7.2443   ( 0.5819   )

Other parameters:
  1.  eval outdegree (density)      -2.4141   ( 0.0389   )   0.0274
  2.  eval reciprocity               2.7108   ( 0.0831   )   0.0179

Overall maximum convergence ratio:    0.0280


Total of 3558 iteration steps.
```

In model 3, the outdegree (density) is -2.4141, and the overall maximum convergence ratio is 0.0280. Due to the negative outdegree, students do not tend to be friends with students that live in the same neighborhood (living nearby).

All out-degrees are negative, which indicates a general tendency against forming ties, which is typical in social networks (to avoid too many connections). All of the estimations of reciprocity are around 2.7. This positive value suggests a strong tendency towards reciprocal relationships, meaning if student A nominates student B as a friend, student B is likely to nominate student A in return.

These ratios are used to assess the convergence of the model. They should ideally be below 0.1 for the model to be considered well-converged.

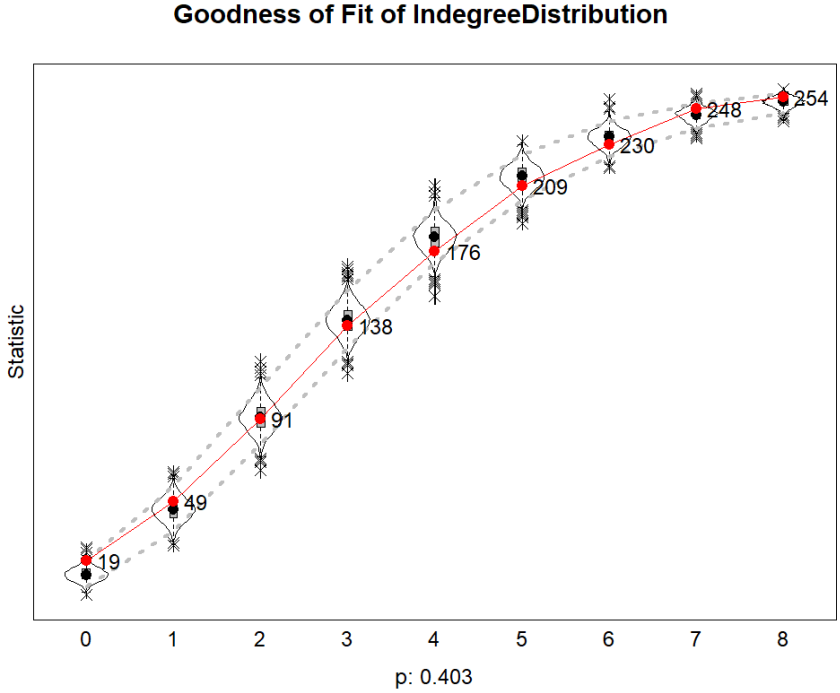
In our models, the maximum convergence ratio is 0.0562, which is below 0.1. This indicates good convergence of our model.

$$\text{t-conv.max} \leq 0.2 \text{ and } \max_k\{|\text{t-conv}_k|\} \leq 0.1$$

(1.4) Are the hypotheses i.-iii. supported by the data? Argue for your answer.
To test the goodness of fit, we calculated the Geodesic Distance and plotted the results.

```
# use parallel computation
cl <- makeCluster(4) # with 4 workers
# Indegree distribution
gofEvId <- sienaGOF(
  model1,
  verbose = FALSE,
  varName = "friendshipNet", IndegreeDistribution,
  cluster = cl
)
# Outdegree distribution
gofEvOd <- sienaGOF(
  model1,
  verbose = FALSE,
  varName = "friendshipNet", OutdegreeDistribution,
  cluster = cl
)
# Triad census
gofEvTC <- sienaGOF(
  model1,
  verbose = FALSE,
  varName = "friendshipNet", TriadCensus,
  cluster = cl
```

The p-value is 0.403 which shows that our model fits very well.



Thus, the hypotheses i to iii can be supported by the data and our model.

(2) We now investigate the co-evolution of friendship (network dependent variable) and alcohol consumption (behavioral dependent variable).
(2.1-2.2)
iv. Popular students tend to increase or maintain their level of alcohol consumption.
First, in order to study the level of the alcohol consumption, the alcohol consumption is now treated as a dependent variable. To test the first hypothesis, we can reorganize our data. The data information is as follows.

```
# Question 2
# Popular students tend to increase or maintain their level of alcohol consumption
# The alcohol consumption is now treated as a dependent variable
alcoholconsumption <- sienaDependent(alcoholCovar, type = "behavior")

my_newdata <- sienaDataCreate(friendshipNet,
                              alcoholconsumption,
                              genderCovar,
                              ageCovar,
                              alcoholCovar,
                              logdistanceCovar)
my_newdata

print01Report(my_newdata, modelname = "knecht_init")

my_neweff <- getEffects(my_newdata)
```

```
> my_newdata
Dependent variables:  friendshipNet, alcoholconsumption
Number of observations: 3

Nodeset                  Actors
Number of nodes             129

Dependent variable friendshipNet
Type               oneMode
Observations       3
Nodeset            Actors
Densities          0.027 0.027 0.028

Dependent variable alcoholconsumption
Type               behavior
Observations       3
Nodeset            Actors
Range              0 - 5

Constant covariates:  genderCovar, ageCovar
Changing covariates:  alcoholCovar
Constant dyadic covariates:  logdistanceCovar
```

We let the “alcoholCovar” be the interaction and run the model.

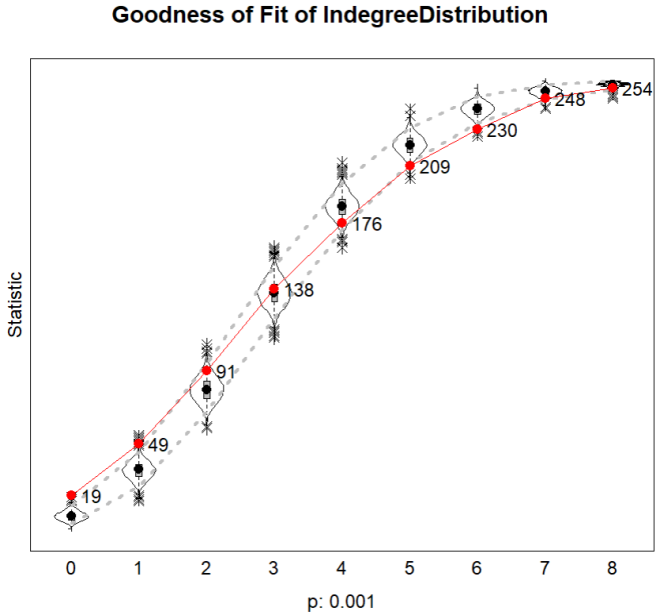
```r
my_neweff <- getEffects(my_newdata)

myeff <- includeEffects(
  my_neweff, outdeg, indeg, avSim,
  name = "alcoholconsumption", interaction1 = "alcoholCovar"
)

effectsDocumentation(myeff)

myAlgorithm <- sienaAlgorithmCreate(
  projname = "CoevKnecht",
  nsub = 4, n3 = 3000, seed = 1908
)
# Model estimation
model.coev <- siena07(myAlgorithm,
                      data = my_newdata, effects = myeff,
                      returnDeps = TRUE, useCluster = TRUE, nbrNodes = 4, prevAns = TRUE
)
model.coev
```

The information of our model is below.

```
> model.coev
Estimates, standard errors and convergence t-ratios

                                                   Estimate   Standard   Convergence
                                                              Error      t-ratio

Network Dynamics
  1. rate constant friendshipNet rate (period 1)  8.6586  ( 0.6651   )    0.0451
  2. rate constant friendshipNet rate (period 2)  7.2157  ( 0.5627   )   -0.0843
  3. eval outdegree (density)                    -2.4074  ( 0.0387   )   -0.0054
  4. eval reciprocity                             2.6983  ( 0.0826   )   -0.0049

Behavior Dynamics
  5. rate rate alcoholconsumption (period 1)      1.4133  ( 0.2345   )   -0.0454
  6. rate rate alcoholconsumption (period 2)      2.1364  ( 0.3464   )    0.0067
  7. eval alcoholconsumption linear shape         0.3781  ( 0.0783   )    0.0184
  8. eval alcoholconsumption quadratic shape     -0.1727  ( 0.0410   )   -0.0226

Overall maximum convergence ratio:    0.1155


Total of 3967 iteration steps.
```

As seen above, the maximum convergence ratio is 0.1155 which is lower than 0.2, thus our model is meaningful.

We tested the goodness of fit and here is the result.



Due to the p equal to 0.001, we cannot derive the conclusion from our model that popular students tend to increase or maintain their level of alcohol consumption.

v. Students tend to adjust their alcohol consumption to that of their friends.

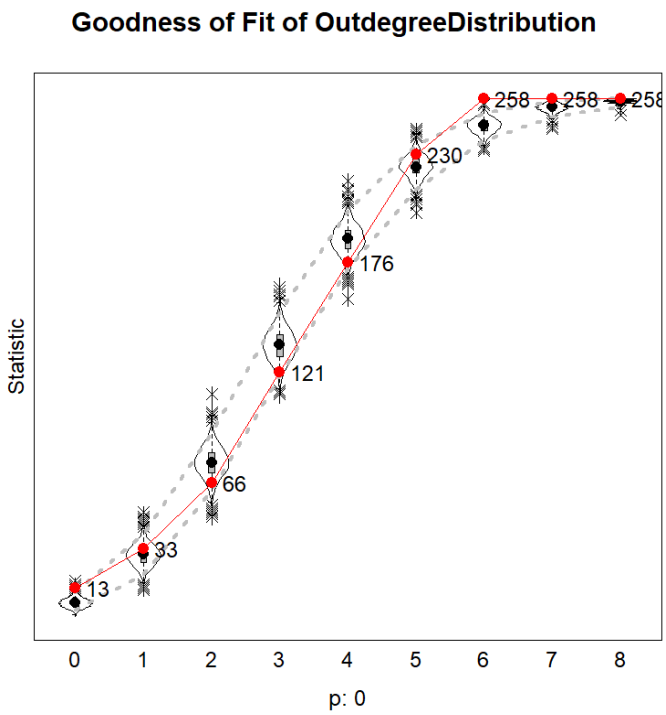
Then we do model estimation. The parameters are the same as the answers in Q1.

```
print01Report(my_newdata, modelname = "knecht_init")

my_neweff <- getEffects(my_newdata)

myeff <- includeEffects(
  my_neweff, outdeg, indeg, avSim,
  name = "alcoholconsumption", interaction1 = "friendshipNet"
)

effectsDocumentation(myeff)

myAlgorithm <- sienaAlgorithmCreate(
  projname = "CoevKnecht",
  nsub = 4, n3 = 3000, seed = 1908
)
# Model estimation
model.coev <- siena07(myAlgorithm,
                      data = my_newdata, effects = myeff,
                      returnDeps = TRUE, useCluster = TRUE, nbrNodes = 4, prevAns = TRUE
)
```

The results are as follows. As seen, the maximum convergence ratio is 0.0494 which is lower than 0.2, thus our model is meaningful.

```
> model.coev
Estimates, standard errors and convergence t-ratios

                                                   Estimate   Standard   Convergence
                                                              Error      t-ratio

Network Dynamics
   1. rate constant friendshipNet rate (period 1)  8.6003  ( 0.6395   )    0.0109
   2. rate constant friendshipNet rate (period 2)  7.2479  ( 0.5650   )   -0.0078
   3. eval outdegree (density)                    -2.4071  ( 0.0387   )    0.0115
   4. eval reciprocity                             2.6981  ( 0.0826   )   -0.0128

Behavior Dynamics
   5. rate rate alcoholconsumption (period 1)      1.5734  ( 0.2423   )    0.0117
   6. rate rate alcoholconsumption (period 2)      2.1951  ( 0.3662   )    0.0088
   7. eval alcoholconsumption linear shape         0.3465  ( 0.3484   )    0.0019
   8. eval alcoholconsumption quadratic shape      0.0634  ( 0.0784   )    0.0054
   9. eval alcoholconsumption average similarity   7.1352  ( 2.3330   )   -0.0149
  10. eval alcoholconsumption indegree             0.1623  ( 0.1754   )    0.0000
  11. eval alcoholconsumption outdegree           -0.1376  ( 0.2065   )   -0.0016

Overall maximum convergence ratio:    0.0494


Total of 4158 iteration steps.
```

We used the same way to check the goodness of fit. Due to the p equal to 0, we cannot derive the conclusion that students tend to adjust their alcohol consumption to that of their friends.



The p-value is 0 which means that the model fits the data badly.

(2.3)

According to our experiments, we found that the hypotheses in the second question are all invalid. From the figures, we can see that our model fits the data fairly; however, the p is around 0.

Hypotheses in (1) focus on how students select their friends. They involve factors such as popularity, similar alcohol consumption habits, and living in the same neighborhood. Since these hypotheses are found to be correct, it indicates evidence of “selection processes”. Selection processes refer to the tendency of individuals to associate with others who are similar to them (homophily) or have certain desirable characteristics (like popularity). Hypotheses in (2) are related to how students may adjust their behavior (specifically, alcohol consumption) in response to the behavior of their peers. This is indicative of “influence processes”, where individuals change their attitudes, beliefs, or behaviors to be more in line with those of their peers. However, since these hypotheses are found to be invalid, there is no evidence supporting influence processes in this context.

Therefore, based on the validation of the hypotheses through SAOMs, the validation of the first set of hypotheses provides clear evidence for selection processes in the formation of friendships among students. This suggests that students do not tend to form friendships based on similar characteristics or desirable traits like popularity. The invalidation of the second set of hypotheses suggests that there is no evidence to support the idea that students significantly adjust their alcohol consumption to align with that of

their friends. This implies that while students may select friends based on certain criteria, their behavior (specifically alcohol consumption) is not significantly influenced by their friends.

(3)

To improve the model in (2) that currently does not support the hypotheses regarding influence processes, particularly in the context of alcohol consumption among students, we can consider incorporating new effects that better account for geodesic distances and degree distributions. We can add a parameter that weighs the influence of peers based on their geodesic distance, with closer peers having a stronger influence.