Group_14_Assignment_2

December 3, 2024

1 Authors: Riccardo Ghetti, Egor Kravchenko, Huijiao Luo

1.1 Task 1: SAOM Modeling Assumptions

Consider the assumptions of SAOMs:

1. The network-behaviour panel data are the outcome of a continuous-time Markov chain.
2. Actors control their outgoing ties and behaviour.
3. At each step: only one tie can change, or the behaviour can increase or decrease by one level.
4. Actors have full knowledge of the network and behaviour.

Critically examine and discuss the plausibility of the assumptions. For each assumption, provide one example of the co-evolution of a network and an actor-level attribute for which the assumption is a reasonable simplification and one example for which the assumption is not tenable. You can use the same example to illustrate the plausibility of more than one assumption.

Solution:

(1) The network-behaviour panel data are the outcome of a continuous-time Markov chain:

The Markovian assumption postulates that future events are conditionally independent from past events given the present, but this can be an oversimplification of certain networks like friendship because actors have memories of the past and are influenced by it regardless of the present state. For example, certain arguments don't lead to an immediate fall out of the friendship, however discontent can stack up over time and influence future decisions. Hence, such a network violates the assumption.

The assumption can be reasonable for the co-evolution processes such as Epidemic Spreading since the states (susceptible, infected, and recovered) of the people strongly depend on the current conditions, especially the number of infected and susceptible individuals nearby.

(2) Actors control their outgoing ties and behaviour

In this case, friendship is reasonably modelled by SAOMs, because a (sane) person has the freedom to choose which people to befriend at any point of time and how to behave.

However, this assumption doesn't work for hierarchical ties like the structure of a company. Indeed, in this case not all actors can choose who is their superior, but it is forcefully assigned to them.

(3) At each step: only one tie can change, or the behaviour can increase or decrease by one level.

This does not hold for very connected and small networks because usually any action or event influences more than just one tie between two actors.

For instance, in friendships completely falling out with someone usually means not being in friendly terms with their friends too.

However, this assumption can hold for the union network in a gambling game since each step only one actor can decide to attack or help any another actor. To be a winner, every actor has to consider all possible interactions and seek to maximize profits rather than acting emotionally.

(4) Actors have full knowledge of the network and behaviour

The assumption seems reasonable for adequately small networks but fails to work when it becomes excessively large or very sparse because it is very possible for information to not reach all actors.

For example, in a small team work network with 3-4 students, it is easy for everyone to know well about each other (including background, time schedule, interaction times) since they need to contact each other frequently for the assignments.

However, it is quite difficult for everyone in the whole class consisting 200 students to have full knowledge of each other, especially most students only meet each other but never talk together.

1.2 Task 2: Network evaluation function

Consider a SAOM with objective function specified by the following statistics:

$$f(i,x,\beta) = \beta_1 s_{1i}(x) + \beta_2 s_{2i}(x) + \beta_3 s_{3i}(x) + \beta_4 s_{4i}(x) + \beta_5 s_{5i}(x,v)$$

with $s_{1i}(x)$ the out-degree (density), $s_{2i}(x)$ the reciprocity, $s_{3i}(x)$ the transitive reciprocated triplets (the reciprocated tie is the tie $i \leftrightarrow j$), $s_{4i}(x)$ the indegree popularity and $s_{5i}(x,v)$ the same covariate effects. x_{ij} denotes the presence or absence of a tie between actors i and j, and v_i denotes the covariate value for actor i.

Question 1: Give the mathematical formula for each effect.

Hint: A useful resource is the RSiena manual.

Solution:

(1) out-degree effect or density effect (density), defined by the out-degree

$$s_{i1}(x) = x_{i+} = \sum_j x_{ij}$$

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

(2) reciprocity effect (recip), defined by the number of reciprocated ties

$$s_{i2}(x) = \sum_j x_{ij} x_{ji}$$

(3)transitive triplets effect (transTrip), defined by the number of transitive patterns in i's relations (ordered pairs of actors (j,h) to both of whom i is tied, while also j is tied to h), for directed networks,

$$s_{i3}(x) = \sum_{j,h} x_{ij} x_{ji} x_{ih} x_{jh}$$

(4) in-degree related popularity effect (inPop) (earlier called popularity or popularity of alter effect), defined by the sum of the in-degrees of the others to whom i is tied,

$$s_{i4}(x) = \sum_j x_{ij} x_{+j} = \sum_j x_{ij} \sum_h x_{hj}$$

(5) covariate-alter or covariate-related popularity (altX), defined by the sum of the covariate over all actors to whom i has a tie,

$$s_{i5}(x) = \sum_j x_{ij} v_j$$

Question 2: Given the current state of the network, with the colour of the nodes representing a binary attribute taking categories 1 (white) and 2 (gray),

Please refer the picture in the assignment document!

and given that $\beta_{1i} = -1.2, \beta_{2i} = 1.5, \beta_{3i} = 1, \beta_{4i} = 0.5$ and $\beta_{5i} = 1.3$, what is the probability that in the next mini-step:

[ ]:
```
# Parameter Setting
b1=-1.2
b2=1.5
b3=1
b4=0.5
b5=1.3
```

i. actor c adds a tie to b ?

c->i	p
c->a	0.087
c->b	0.712
c->c	0.071
c->d	0.130

[ ]:
```
# actor c adds a tie to b
f_cb = b1*2+b2*1+b3+4*b4
exp_f_cb = exp(f_cb)

# actor c delete actor a
f_ca = 0
exp_f_ca = exp(f_ca)

# actor c adds a tie to d
f_cd= b1*2+b4*3+b5*1
exp_f_cd = exp(f_cd)
```

```
# actor c keeps unchanged
f_cc = 1*b1+2*b4
exp_f_cc = exp(f_cc)

# Sum up the exp(fi)
SumExp_fc = exp_f_cb + exp_f_ca + exp_f_cd + exp_f_cc

# Calculate the probability of actor c adding a tie to b
p1=exp_f_cb/SumExp_fc
p1
```

0.711541807795676

ii. actor b adds a tie to actor a ?

b->i	p
b->a	0.900
b->b	0.015
b->c	0.030
b->d	0.055

[ ]:
```
# actor b adds a tie to a
f_ba = 2*b1+b2+b3+4*b4+b5
exp_f_ba = exp(f_ba)

# actor b deletes the tie to c
f_bc = 0
exp_f_bc = exp(0)

# actor b adds a tie to d
f_bd = 2*b1+b2+3*b4
exp_f_bd = exp(f_bd)

# actor b keeps unchanged
f_bb = b1+b4
exp_f_bb = exp(f_bb)

# Sum up the exp(fi) for actor b
SumExp_fb = exp_f_ba + exp_f_bb + exp_f_bc + exp_f_bd

# Calculate the probability of actor b adding a tie to a
p2=exp_f_ba/SumExp_fb
p2
```

0.900287725485279

iii. actor a deletes the tie to b ?

a->i	p
a->a	0.250
a->b	0.083
a->c	0.555
a->d	0.112

[ ]:
```
# actor a deletes the tie to b
f_ab = b1+b2+b4
exp_f_ab = exp(f_ab)

# actor a deletes the tie to d
f_ad = b1+2*b4+b5
exp_f_ad = exp(f_ad)

# actor a adds a tie to c
f_ac = 3*b1+b2+b3+5*b4+b5
exp_f_ac = exp(f_ac)

# actor a keeps unchanged
f_aa = 2*b1+b2+3*b4+b5
exp_f_aa = exp(f_aa)

# Sum up exp(fi) for actor a
SumExp_fa = exp_f_aa + exp_f_ab + exp_f_ac + exp_f_ad

# Calculate the probability of actor a deleting the tie to b
p3=exp_f_ab / SumExp_fa
p3
```

0.0830570352889286

iv. actor d does not change anything?

d->i	p
d->a	0.026
d->b	0.114
d->c	0.766
d->d	0.094

[ ]:
```
# actor d keeps unchanged
f_dd = 2*b1+b2+4*b4
exp_f_dd = exp(f_dd)

# actor d adds c
f_dc = 3*b1+b2+b3+6*b4+b5
exp_f_dc = exp(f_dc)
```

```
# actor d deletes a
f_da = b1+2*b4
exp_f_da = exp(f_da)

# actor d deletes b
f_db = b1+b2+2*b4
exp_f_db = exp(f_db)

# Sum up exp(fi) for actor d
SumExp_fd = exp_f_da + exp_f_db + exp_f_dc + exp_f_dd

p4= exp_f_dd / SumExp_fd
p4
```

0.0938077111157231

1.3 Task 3: Simulations from SAOM

The file *simSAOM.R* contains the code to simulate the network evolution between two observations from a SAOM with an evaluation function specified by outdegree, reciprocity and dyadic covariate 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.

[ ]:
```
install.packages("RSiena")
install.packages("sna")
install.packages("parallel")
install.packages("MASS")

library(RSiena)
library(sna)
library(parallel)
library(MASS)
library(network)
library(statnet.common)
```

[ ]:
```
comp_stat<-function(x1,W,i,mean_W,n){
  a=0
  b=0
```

```
  d=0
  for (k in 1:n){
    a=a+x1[i,k]  #outdegree
    b=b+x1[i,k]*x1[k,i]  #reciprocity
    d=d+x1[i,k]*(W[i,k]-mean_W)     #dyadic covariate centered effect
  }
  return(as.double(c(a,b,d)))
}

comp_prob<-function(b1,b2,b3,vect_stat,n){
  vect_prob=matrix(0,1,n)
  sum=0
  #compute the normalizer constant and save the weights
  for (k in 1:n)  {
    p=exp(b1*vect_stat[1,k]+b2*vect_stat[2,k]+b3*vect_stat[3,k])
    sum=sum+p
    vect_prob[k]=p
    }

  #normalize the probabilities
  for (k in 1:n)
    vect_prob[k]=vect_prob[k]/sum
  return(vect_prob)
}
```

[ ]:
```
# Task 3.1 ----------------------------------------------------------------
# The function "simulation" simulates the network evolution between
# two time points.
# Given the network at time t1, denoted by x1, the function simulates the
# steps of the continuous-time Markov chain defined by a SAOM with outdegree,
# recip and dyadic covariate (W matrix) statistics.
# Unconditional simulation is used.
# The function returns the network at time t2.
# The structure of the algorithm is described in the file
# _Simulating from SAOM.pdf_ available in
# the Lecture notes and additional material section on Moodle.

#' Simulate the network evolution between two time points
#'
#' @param n number of actors in the network
#' @param x1 network at time t1
#' @param W matrix of dyadic covariate
#' @param lambda rate parameter
#' @param beta1 outdegree parameter
#' @param beta2 reciprocity parameter
#' @param beta3 dyadic covariate parameter
#'
```

```r
#' @return network at time t2

simulation <- function(n, x1, W, lambda, beta1, beta2, beta3) {
  t <- 0    # time
  x <- x1
  mean_W=mean(matrix(W,1,n*n))
  while (t < 1) {
    dt <- rexp(1, n * lambda)
    #draw a random actor with uniform prob from the n available
    i=sample(1:n,size=1)
    vect_stat=matrix(0,3,n)  #initialize the container of all statistics

    #computing the statistics for all choices
    for (k in 1:n){
      #if k==i we interpret it as the choice of leaving the network unchanged
      if(k==i){
        #Break is necessary to stop the computation in this iteration
        vect_stat[1:3,k]=comp_stat(x1,W,k,mean_W,n)
        break
      }

      #copy the current network for computing statistics of its variation
      x2=x
      #invert the tie i->j in the hypothetical network
      x2[i,k]=xor(x2[i,k],1)
      #compute and store the statistics for the current choice
      vect_stat[1:3,k]=comp_stat(x2,W,k,mean_W,n)

    }

    #compute the multinomial probabilities for every choice
    vect_prob=comp_prob(beta1,beta2,beta3,vect_stat,n)
    #sample the actor according to the computed probabilities
    j=sample(x=1:n,size=1,prob=vect_prob)

    if (i!=j)    #invert the tie i->j in x else do nothing
      x[i,j]=xor(x[i,j],1)
    t=t+dt
   }

  return(x)
}
```

(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. Additionally, the dyadic covariate is given in the file *W.csv*. Estimate the parameters of the SAOM with outdegree, reciprocity and dyadic covariate effects statistics using the function ***siena07***.

[ ]:

```r
# Task 3.2 ----------------------------------------------------------------
# Consider the two adjacency matrices in the files net1.csv and net2.csv.
# Estimate the parameters of the SAOM with outdegree, reciprocity and
# dyadic covariate statistics using the function `siena07`.
# You can extract the estimated parameters from the `rate` and `theta`
#  components of the output object (e.g., res$rate and res$theta).

# ---MISSING---
#setwd("C:/Users/Riccardo Ghetti/Downloads/Network modeling/Assignment2/
 ↪Assignment2")
net1=as.matrix(read.csv("net1.csv",header=FALSE))
net2=as.matrix(read.csv("net2.csv",header=FALSE))
W=as.matrix(read.csv("W.csv",header=FALSE))

friendship=sienaDependent(array(c(net1,net2), dim=c(22,22,2)))

dyad=coDyadCovar(W)
mydata=sienaDataCreate(friendship,dyad)
mydata

myeff=getEffects(mydata)
myeff
effectsDocumentation(myeff)

myeff=includeEffects(myeff,X,interaction1 = "dyad")

myAlgorithm <- sienaAlgorithmCreate(
  projname = "friends_W",
  nsub = 4, n3 = 3000, seed = 1908
)

model0 <- siena07(
  myAlgorithm,
  data = mydata, effects = myeff,
  returnDeps = TRUE,
  useCluster = TRUE, nbrNodes = 4, batch = FALSE
)
rate=model0$rate
betas=model0$theta
rm(model0)
```

```
Dependent variables:  friendship
Number of observations: 2

Nodeset                  Actors
Number of nodes              22
```

```
Dependent variable friendship
Type               oneMode
Observations       2
Nodeset            Actors
Densities          0.17 0.28

Constant dyadic covariates:  dyad
```

A sienaEffects: 292 × 31

	name	effectName
	<chr>	<chr>
friendship.rate.1	friendship	basic rate parameter friendship
friendship.rate.2	friendship	outdegree effect on rate friendship
friendship.rate.3	friendship	indegree effect on rate friendship
friendship.rate.4	friendship	reciprocity effect on rate friendship
friendship.rate.5	friendship	effect 1/outdegree on rate friendship
friendship.rate.6	friendship	effect ln(outdegree+1) on rate friendship
friendship.rate.7	friendship	effect 1/indegree on rate friendship
friendship.rate.8	friendship	effect ln(indegree+1) on rate friendship
friendship.rate.9	friendship	effect 1/reciprocity on rate friendship
friendship.rate.10	friendship	effect ln(reciprocity+1) on rate friendship
friendship.obj.eval.1	friendship	outdegree (density)
friendship.obj.endow.1	friendship	outdegree (density)
friendship.obj.creation.1	friendship	outdegree (density)
friendship.obj.eval.2	friendship	reciprocity
friendship.obj.endow.2	friendship	reciprocity
friendship.obj.creation.2	friendship	reciprocity
friendship.obj.gmm.1	friendship	new recip.
friendship.obj.gmm.2	friendship	persistent recip.
friendship.obj.gmm.3	friendship	real recip.
friendship.obj.eval.3	friendship	transitive triplets
friendship.obj.endow.3	friendship	transitive triplets
friendship.obj.creation.3	friendship	transitive triplets
friendship.obj.eval.4	friendship	transitive triplets (1)
friendship.obj.endow.4	friendship	transitive triplets (1)
friendship.obj.creation.4	friendship	transitive triplets (1)
friendship.obj.eval.5	friendship	transitive triplets (2)
friendship.obj.endow.5	friendship	transitive triplets (2)
friendship.obj.creation.5	friendship	transitive triplets (2)
friendship.obj.eval.6	friendship	transitive mediated triplets
friendship.obj.endow.6	friendship	transitive mediated triplets
friendship.obj.eval.92	friendship	unspecified interaction effect
friendship.obj.endow.79	friendship	unspecified interaction effect
friendship.obj.creation.79	friendship	unspecified interaction effect
friendship.obj.eval.93	friendship	unspecified interaction effect
friendship.obj.endow.80	friendship	unspecified interaction effect
friendship.obj.creation.80	friendship	unspecified interaction effect
friendship.obj.eval.94	friendship	unspecified interaction effect
friendship.obj.endow.81	friendship	unspecified interaction effect
friendship.obj.creation.81	friendship	unspecified interaction effect
friendship.obj.eval.95	friendship	unspecified interaction effect
friendship.obj.endow.82	friendship	unspecified interaction effect
friendship.obj.creation.82	friendship	unspecified interaction effect
friendship.obj.eval.96	friendship	unspecified interaction effect
friendship.obj.endow.83	friendship	unspecified interaction effect
friendship.obj.creation.83	friendship	unspecified interaction effect
friendship.obj.eval.97	friendship	unspecified interaction effect
friendship.obj.endow.84	friendship	unspecified interaction effect
friendship.obj.creation.84	friendship	unspecified interaction effect
friendship.obj.eval.98	friendship	unspecified interaction effect
friendship.obj.endow.85	friendship	unspecified interaction effect

```
Effects documentation written to file myeff.html .
  effectName include fix   test  initialValue parm
1 dyad       TRUE    FALSE FALSE          0   0
If you use this algorithm object, siena07 will create/use an output file
Siena.txt .

Warning message:
"no DISPLAY variable so Tk is not available"
No X11 device available, forcing use of batch mode



Start phase 0
theta: 0.861 0.000 0.000

Start phase 1
Phase 1 Iteration 1 Progress: 0%
Phase 1 Iteration 5 Progress: 0%
Phase 1 Iteration 25 Progress: 0%
Phase 1 Iteration 45 Progress: 0%
theta: 0.7959 0.1344 0.0736

Start phase 2.1
Phase 2 Subphase 1 Iteration 1 Progress: 5%
Phase 2 Subphase 1 Iteration 2 Progress: 5%
theta 0.716 0.263 0.175
ac 0.622 1.740 2.590
Phase 2 Subphase 1 Iteration 3 Progress: 5%
Phase 2 Subphase 1 Iteration 4 Progress: 5%
theta 0.661 0.405 0.302
ac 1.314 0.566 2.149
Phase 2 Subphase 1 Iteration 5 Progress: 5%
Phase 2 Subphase 1 Iteration 6 Progress: 5%
theta 0.840 0.311 0.311
ac 0.769 0.178 1.788
Phase 2 Subphase 1 Iteration 7 Progress: 5%
Phase 2 Subphase 1 Iteration 8 Progress: 5%
theta 0.955 0.182 0.285
ac 0.734 0.177 1.791
Phase 2 Subphase 1 Iteration 9 Progress: 5%
Phase 2 Subphase 1 Iteration 10 Progress: 5%
theta 0.941 0.188 0.274
ac 0.567 0.164 1.705
theta 0.820 0.285 0.253
ac -0.330 -0.128  0.867
theta: 0.820 0.285 0.253

Start phase 2.2
Phase 2 Subphase 2 Iteration 1 Progress: 10%
```

```
Phase 2 Subphase 2 Iteration 2 Progress: 10%
Phase 2 Subphase 2 Iteration 3 Progress: 10%
Phase 2 Subphase 2 Iteration 4 Progress: 10%
Phase 2 Subphase 2 Iteration 5 Progress: 10%
Phase 2 Subphase 2 Iteration 6 Progress: 10%
Phase 2 Subphase 2 Iteration 7 Progress: 10%
Phase 2 Subphase 2 Iteration 8 Progress: 10%
Phase 2 Subphase 2 Iteration 9 Progress: 10%
Phase 2 Subphase 2 Iteration 10 Progress: 10%
theta 0.856 0.268 0.255
ac -0.1601 -0.0785 -0.0159
theta: 0.856 0.268 0.255

Start phase 2.3
Phase 2 Subphase 3 Iteration 1 Progress: 15%
Phase 2 Subphase 3 Iteration 2 Progress: 15%
Phase 2 Subphase 3 Iteration 3 Progress: 15%
Phase 2 Subphase 3 Iteration 4 Progress: 15%
Phase 2 Subphase 3 Iteration 5 Progress: 15%
Phase 2 Subphase 3 Iteration 6 Progress: 15%
Phase 2 Subphase 3 Iteration 7 Progress: 15%
Phase 2 Subphase 3 Iteration 8 Progress: 15%
Phase 2 Subphase 3 Iteration 9 Progress: 15%
Phase 2 Subphase 3 Iteration 10 Progress: 15%
theta 0.835 0.276 0.254
ac -0.12437 -0.00937 -0.00824
theta: 0.835 0.276 0.254

Start phase 2.4
Phase 2 Subphase 4 Iteration 1 Progress: 21%
Phase 2 Subphase 4 Iteration 2 Progress: 21%
Phase 2 Subphase 4 Iteration 3 Progress: 21%
Phase 2 Subphase 4 Iteration 4 Progress: 21%
Phase 2 Subphase 4 Iteration 5 Progress: 21%
Phase 2 Subphase 4 Iteration 6 Progress: 21%
Phase 2 Subphase 4 Iteration 7 Progress: 21%
Phase 2 Subphase 4 Iteration 8 Progress: 21%
Phase 2 Subphase 4 Iteration 9 Progress: 22%
Phase 2 Subphase 4 Iteration 10 Progress: 22%
theta 0.840 0.271 0.253
ac -0.0799 -0.0995 -0.0244
theta: 0.840 0.271 0.253

Start phase 3
Phase 3 Iteration 1 Progress 30%
Phase 3 Iteration 801 Progress 49%
Phase 3 Iteration 1601 Progress 68%
Phase 3 Iteration 2401 Progress 86%
```

(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.

[ ]:
```
# Task 3.3 -----------------------------------------------------------------
# Conditioning on the first observation, generate 1,000 simulations of the
# network evolution
# Compute the triad census counts for each simulated network.
# Save the results in an object, named `triadCensus`, in which rows are
# the index of a simulated network and columns are the type of triads.
# Column names should use the triad type name, e.g., "003", "012", "102", ...

results=array(0,dim=c(1000,22,22))
beta1=betas[1]
beta2=betas[2]
beta3=betas[3]

for (i in 1:1000){
  results[i,1:22,1:22]=simulation(22,net1,W,rate,beta1,beta2,beta3)
}
triadCensus=triad.census(results)
```

(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. Standardized the simulated network stats.

[ ]:
```
# Task 3.4 -----------------------------------------------------------------
## i. standardized the simulated network stats. ----
##   Name the resulting object as triadCensusStd

triad_stat=matrix(0,16,2)

for (e in 1:16){
  triad_stat[e,1]=mean(triadCensus[1:1000,e]) #the mean of each column
  triad_stat[e,2]=sd(triadCensus[1:1000,e]) #its standard deviation
}

triadCensusStd=matrix(0,1000,16)
```

```r
for(e in 1:16){
  average_e=triad_stat[e,1]
  sd_e=triad_stat[e,2]
  for (g in 1:1000){
    triadCensusStd[g,e]=(triadCensus[g,e]-average_e)/sd_e  #standardize the values for each network
  }
}
```

ii. Variance-covariance matrix and its generalized inverse.

[ ]:
```r
## ii. variance-covariance matrix and its generalized inverse.        ----

p_h=cov(triadCensusStd,triadCensusStd)  #compute the covariance matrix
p_inv=ginv(p_h)                         #compute its generalized inverse
```

iii. Standardized the observed values of the triad census counts.

[ ]:
```r
## iii. standardized the observed values of the triad census counts    ----
##  in the second observation using values from i.

net2

Obs_stat=triad.census(net2)
for (e in 1:16){
  Obs_stat[e]=(Obs_stat[e]-triad_stat[e,1])/triad_stat[e,2] #normalize obs stat
}
```

A matrix: 22×22 of type int

V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V13	V14	V15	V16
0	0	1	1	0	1	0	1	0	1	1	0	0	0
0	0	0	0	0	0	1	1	0	1	0	0	0	0
1	1	0	0	0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	1	0	1
0	0	1	0	0	0	0	1	0	0	0	0	1	0
1	1	1	0	0	0	0	1	0	1	0	0	0	0
0	1	0	1	0	1	0	0	0	1	0	0	0	0
0	1	1	0	0	1	1	1	0	1	0	0	1	0
0	1	0	0	0	0	0	1	0	0	0	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	1	1	1	0	1	1	1	0	1	0	0	0	0
0	1	0	1	0	0	0	1	0	1	0	0	0	0
1	0	0	1	1	0	1	1	0	0	1	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	1	0	1	1	1	0	1	1	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0	0	0
0	1	0	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1	0	1	0	0	0	0

iv. Monte-Carlo Mahalanobis distance computation.

[ ]:
```r
## iv. Monte-Carlo Mahalanobis distance computation
    ----
# Compute the Mahalanobis distance using the mhd function for
# the auxiliar statistics of the simulated networks and the observed network.
# Remember to drop statistics with variance of 0 for the plot and
# Mahalanobis distance computation, report which statistics suffer this issue.

#' Compute the Mahalanobis distance
#'
#' @param auxStats numerical vector with the mean centered or standardized
#'   auxiliar statistics
#' @param invCov numerical matrix with the inverse of the variance-covariance
#'   matrix of the auxiliar statistics in the simulated networks
#'
#' @return numeric value with the Mahalanobis distance of auxiliar stats
#'
#' @examples
#' mhd(c(2, 4) - c(1.5, 2), solve(matrix(c(1, 0.8, 0.8, 1), ncol = 2)))
mhd <- function(auxStats, invCov) {
  t(auxStats) %*% invCov %*% auxStats
  }
```

```r
mahal=matrix(0,1001,1)

for(e in 1:1000){
  mahal[e]=mhd(triadCensusStd[e,1:16], p_inv)
}

mahal[1001]=mhd(t(Obs_stat), p_inv)
```

v. Monte-Carlo p-value computation.

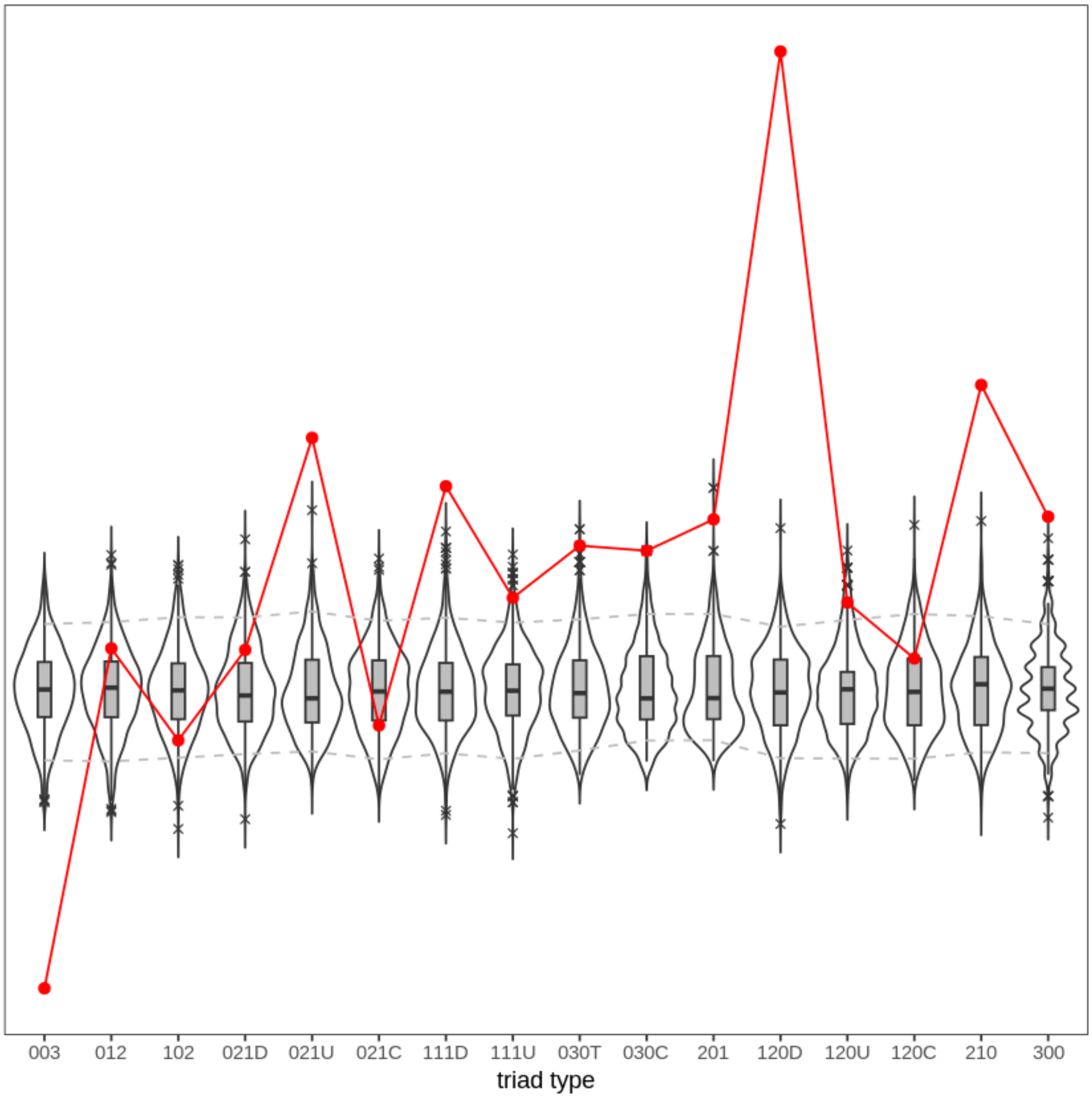
[ ]:
```r
## v. Monte-Carlo p-value computation                                ----
# Compute the proportion of simulated networks where the distance is
# equal or greater than the distance in the observed network.

p_val=0
for (e in 1:1000){
  if(mahal[e]>=mahal[1001])
    p_val=p_val+1
}
p_val=p_val/1000
p_val

# violin plots ------------------------------------------------------------
# Fill out the missing part and run the code to obtain the violin plots

# install.packages(c("tidyverse", "ggplot2"))  # # run this line to install
library(tidyverse)  # used: dplyr and tidyr
library(ggplot2)
colnames(triadCensusStd)<-colnames(triadCensus)

# Given the array triadCensusStd, create a data frame from it in a long format,
# do the same for the observed network statistics at time t2.
# Named the data frame "triadCensusDf" and "triadCensusObs".
# Drops statistics with variance of 0 for the plot.

triadCensusDf <- data.frame(triadCensusStd) |>
  select(where(~ var(.) > 0)) |>  # Drop statistics with zero variance
  pivot_longer(
    everything(),
    names_to = "triad", names_pattern = "^X(.+)$",
    values_to = "nnodes"
  )

# Compute the statistics of the observed network at time t2,
#  standardized using the stats from 2.4 literal i.
obs=data.frame(Obs_stat)
triadCensusObs <- obs |>
```

```r
  pivot_longer(
    everything(),
    names_to = "triad", names_pattern = "^X(.+)$",
    values_to = "nnodes"
  ) |>
  filter(triad %in% unique(triadCensusDf$triad))

# The following code computes the 5% and the 95% quantiles
# of the triad counts by type
percTriad <- triadCensusDf |>
  group_by(triad) |>
  summarise(
    quant05 = quantile(nnodes, prob = 0.05),
    quant95 = quantile(nnodes, prob = 0.95)
  ) |>
  pivot_longer(
    starts_with("quant"),
    names_to = "quant", names_pattern = "quant(.+)",
    values_to = "nnodes"
  )

plot.new()
# The following code produces the violin plots
ggplot(triadCensusDf, aes(fct_inorder(triad), nnodes)) +
  geom_violin(trim = FALSE, scale = "width") +
  stat_summary(fun = mean, geom = "point", size = 2) +
  geom_boxplot(width = 0.2, fill = "gray", outlier.shape = 4) +
  geom_point(data = triadCensusObs, col = "red", size = 2) +
  geom_line(
    data = triadCensusObs, aes(group = 1), col = "red", linewidth = 0.5
  ) +
  geom_line(
    data = percTriad, mapping = aes(group = quant),
    col = "gray", linetype = "dashed"
  ) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.title.y = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks.y = element_blank()
  ) +
  xlab("triad type") +
  title("Goodness of Fit of Triad Census Counts")
```

0

```
  Attaching core tidyverse packages               tidyverse
2.0.0
 dplyr     1.1.4       readr     2.1.5
 forcats   1.0.0       stringr   1.5.1
 ggplot2   3.5.1       tibble    3.2.1
 lubridate 1.9.3       tidyr     1.3.1
 purrr     1.0.2
  Conflicts
tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
 dplyr::select() masks MASS::select()
 Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

Goodness of Fit of Triad Census Counts



1.3.1 Conclusion

As it can be seen from the violin plots, the simulated networks for the most part don't have an accurate goodness of fit regarding the triad census. While 5 out of 16 have their observed value within the 5-95% quantile interval, all of the other types show that the observed value in net2 is a very big outlier. This bad goodness of fit is also exacerbated by the p_value of the Mahalanobis distance: 0; indeed it can be seen that the real value is 576 while the simulated ones rarely exceed 20. Therefore our simulations don't model in a satisfactory way the data.

1.4 Task 4: Estimation and interpretation of SAOMs

The folder *Glasgow.zip* contains data collected by Michell and West (1996) under the "Teenage Friends and Lifestyle Study" 2. The dataset was collected on a cohort of 160 students followed over two years starting in February 1995, when the pupils were aged 13, and ending in January 1997. The friendship network of the pupils was observed at three-time points. Pupils were asked to name up to six friends and provide information on their socio-demographic characteristics along with the use of substances, such as tobacco and alcohol consumption. In the following, we analyse the data of the 129 pupils who were present at all three-time points. The folder contains the following files - f1, f2, f3.csv: adjacency matrices of the friendship networks - demographic.csv: data frame containing information on gender (1 boy, 2 girl) and age - logdistance.csv: logarithm of the distance (in kilometers) between the houses of the pupils - alcohol.csv: alcohol consumption coded as 1 (non), 2 (once or twice a year), 3 (once a month), 4 (once a week) and 5 (more than once a week);

We are interested in investigating the co-evolution of friendship (network dependent variable) and alcohol consumption (behavioral dependent variable).

[ ]:
```
source("siena07ToConverge.R")

# Importing the adjacency matrices of the networks
net1 <- as.matrix(read.csv("f1.csv", header = FALSE))
net2 <- as.matrix(read.csv("f2.csv", header = FALSE))
net3 <- as.matrix(read.csv("f3.csv", header = FALSE))

num_pupils <- 129

# Reading the demographic and alcohol consumption characteristics of the␣
↪students
attributes <- read.csv("demographic.csv", header = TRUE)
alcohol <- as.matrix(
  read.csv("alcohol.csv", header = TRUE)
)

# Reading the information on the distance between the houses of the pupils

log_distance <- as.matrix( read.csv("logdistance.csv", header = FALSE))
```

(2.1) Start by computing the Jaccard index to evaluate if the data contains enough information to investigate the evolution of the friendship network. Comment on the results.

[ ]:
```
# Creation of a Siena network object
friendship <- sienaDependent(
  array(c(net1, net2, net3), dim = c(num_pupils, num_pupils, 3))
)



gender <- coCovar(attributes$gender)
age <- coCovar(attributes$age)
```

```
log_dist <- coDyadCovar(log_distance)

#The alcohol consumption is treated as a dependent variable.
alco_beh <- sienaDependent(alcohol, type = "behavior")

mydata <- sienaDataCreate(friendship, gender, age, log_dist, alco_beh)
mydata

## precondition of the analysis
# Data description
# Stability: Jaccard index
print01Report(mydata, modelname = "Glasgow")
```

```
Dependent variables:  friendship, alco_beh
Number of observations: 3

Nodeset                  Actors
Number of nodes             129

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

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

Constant covariates:  gender, age
Constant dyadic covariates:  log_dist
```

Conclusion: Jacard index is 0.304 for observations 1 & 2, and is 0.351 for observations 2 & 3. Both these numbers are greater than 0.3, so data contains enough information to investigate the evolution of the friendship network.

(2.2) Specify a reasonable model to test the following hypotheses:

[ ]:
```
myeff <- getEffects(mydata)
myeff

#effectsDocumentation
```

H1: Students tend to be friends with popular pupils.

Idea: To test the hypothesis H1, we include the in-degree popularity effect, which reflects tendencies to dispersion in in-degrees of the actors; or, tendencies for actors with high in-degrees to attract extra incoming ties 'because' of their high current in-degrees

```
[ ]: #We include  transitive triplets effect to represent the dynamics in local␣
      ↪structure
     myeff <- includeEffects(myeff, transTrip)

     # To test the hypothesis H1,
     #we include the in-degree popularity effect, which reflects tendencies to␣
      ↪dispersion in in-degrees of the actors; or,
     #tendencies for actors with high in-degrees to attract extra incoming ties␣
      ↪'because' of their high current in-degrees
     myeff <- includeEffects(myeff, name = "friendship", inPop)
```

```
  effectName          include fix   test  initialValue parm
1 transitive triplets TRUE    FALSE FALSE            0    0
  effectName            include fix   test  initialValue parm
1 indegree - popularity TRUE    FALSE FALSE            0    0
```

H2: Students tend to be friends with students that live in the same neighborhood (living nearby).

Idea: To test the hypothesis H2, we include the covariate main effect of the logarithm of the distance between the houses of the pupils.

```
[ ]: #To test the hypothesis H2, we include the covariate main effect of the␣
      ↪logarithm of the distance between the houses of the pupils
     myeff <- includeEffects(myeff, X, name = "friendship", interaction1 =␣
      ↪"log_dist")
```

```
  effectName include fix   test  initialValue parm
1 log_dist   TRUE    FALSE FALSE            0    0
```

H3: Popular students tend to increase or maintain their level of alcohol consumption.

Idea: To test the hypothesis H3, we include the indegree effect for behavioral evaluation function.

```
[ ]: #To test the hypothesis H3, we include the indegree effect for behavioral␣
      ↪evaluation function.
     myeff <- includeEffects(myeff, indeg, name = "alco_beh", interaction1 =␣
      ↪"friendship")
```

```
  effectName        include fix   test  initialValue parm
1 alco_beh indegree TRUE    FALSE FALSE            0    0
```

H4: Students tend to adjust their alcohol consumption to that of their friends.

Idea: To test the hypothesis H4, we include the average similarity effect, defined by the average of centered similarity scores (of alcohol consumption) between pupil and the other pupils to whom he is tied.

```
[ ]: #To test the hypothesis H4, we include the average similarity effect, defined␣
      ↪by the average of centered similarity scores
```

```
#(of alcohol consumption) between pupil and the other pupils to whom he is tied.
myeff <- includeEffects(myeff, avSim, name = "alco_beh", interaction1 =
    'friendship')
```

```
  effectName                include fix   test  initialValue parm
1 alco_beh average similarity TRUE    FALSE FALSE            0    0
```

[ ]:
```
myeff
```

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

[ ]:
```
## Model estimation ----------------------------------------------------------
# Specifying the parameter of the algorithm
myAlgorithm <- sienaAlgorithmCreate(
  projname = "friends_res",
  nsub = 4, n3 = 10000, seed = 239239
)

# Estimate the model
model1 <- siena07(
  myAlgorithm,
  data = mydata, effects = myeff,
  returnDeps = TRUE,
  useCluster = TRUE, nbrNodes = 4, batch = FALSE
)

model1


#Check the convergence of the model
t.conv <- apply(model1$sf, 2, mean) / apply(model1$sf, 2, sd)
overall <- sqrt(t(apply(model1$sf, 2, mean)) %*% solve(cov(model1$sf)) %*%
    apply(model1$sf, 2, mean))
#all numbers of t.conv are less than 0.1 by absolute value and the value of
    overall is less than 0.2.
#Thus we can conclude that the model converges well.

#We double check the convergence using siena07ToConvergence function.

  model1 <- siena07(myAlgorithm,
  data = mydata, effects = myeff, returnDeps = TRUE, prevAns = model1,
  useCluster = TRUE, nbrNodes = 4
  )

siena07ToConvergence(myAlgorithm, dat = mydata, eff = myeff)
```

```
#The again see that all t-ratios are less than 0.1 by absolute value and the
↪overall maximum convergence ratio
#is less than 0.2. Therefore, we received another evidence of that the model
↪converges well.

## Check the goodness of fit

# Indegree distribution
gofCoevId <- sienaGOF(
  model1,
  verbose = FALSE,
  varName = "friendship", IndegreeDistribution
)

# Outdegree distribution
gofCoevOd <- sienaGOF(
  model1,
  verbose = FALSE,
  varName = "friendship", OutdegreeDistribution
)

# Triad census
gofCoevTC <- sienaGOF(
  model1,
  verbose = FALSE,
  varName = "friendship", TriadCensus
)



# Behaviour distribution
gofCoevBeh <- sienaGOF(
  model1,
  verbose = FALSE,
  varName = "alco_beh", BehaviorDistribution,
)


plot(gofCoevId)
plot(gofCoevOd)
plot(gofCoevTC, center = TRUE, scale = TRUE)
plot(gofCoevBeh)

descriptives.sienaGOF(gofCoevBeh)
```

```
If you use this algorithm object, siena07 will create/use an output file
Siena.txt .
```

```
No X11 device available, forcing use of batch mode


Start phase 0
theta:  7.454  6.829 -1.613  0.000  0.000  0.000  0.000  0.910  1.093  0.347
0.000  0.000  0.000

Start phase 1
Phase 1 Iteration 1 Progress: 0%
Phase 1 Iteration 5 Progress: 0%
Phase 1 Iteration 25 Progress: 0%
Phase 1 Iteration 45 Progress: 0%
theta:  7.99309  6.88807 -1.69686  0.11292  0.09456  0.00398  0.00843  0.81553
1.09860  0.34417 -0.00756 -1.00000  0.01250

Start phase 2.1
Phase 2 Subphase 1 Iteration 1 Progress: 6%
Phase 2 Subphase 1 Iteration 2 Progress: 6%
theta  8.58860  7.04819 -1.80835  0.38537  0.33760 -0.00489  0.01019  0.74947
1.16516  0.35619 -0.03289 -1.72945  0.02494
ac 1.055 1.751 0.765 1.447 1.931 0.661 1.223 1.062 0.897 1.366 0.990 0.851 1.331
Phase 2 Subphase 1 Iteration 3 Progress: 6%
Phase 2 Subphase 1 Iteration 4 Progress: 6%
theta  8.8665  7.8212 -2.0211  1.1041  0.5267 -0.0257 -0.0448  0.8822  1.3394
0.3678 -0.0270 -0.1561  0.0401
ac 1.525 1.824 1.434 1.396 3.890 1.759 1.192 1.276 0.951 1.384 1.043 0.949 1.349
Phase 2 Subphase 1 Iteration 5 Progress: 6%
Phase 2 Subphase 1 Iteration 6 Progress: 6%
theta  8.935911  8.589820 -2.225025  1.749016  0.585853 -0.043697 -0.103768
1.222308  1.623694  0.335846  0.000335  3.369335  0.041138
ac 1.499 1.526 1.024 0.709 0.386 1.099 1.136 1.318 1.065 1.051 1.009 0.944 1.163
Phase 2 Subphase 1 Iteration 7 Progress: 6%
Phase 2 Subphase 1 Iteration 8 Progress: 6%
theta  9.3053  8.8834 -2.3366  2.1115  0.6000 -0.0536 -0.1361  1.4090  1.7821
0.3047  0.0186  5.3399  0.0478
ac 1.3806 1.4981 0.8009 0.2900 0.0237 0.8110 0.9683 1.3186 1.0720 0.8983 1.0040
0.9037 0.9913
Phase 2 Subphase 1 Iteration 9 Progress: 6%
Phase 2 Subphase 1 Iteration 10 Progress: 6%
theta  9.6674  9.0010 -2.4117  2.3185  0.6121 -0.0615 -0.1584  1.5369  1.9305
0.2632  0.0405  6.8994  0.0570
ac  1.3797  1.3954  0.6888  0.1421 -0.0917  0.6885  0.8708  1.2938  1.0572
0.7231  1.0121  0.9043  0.7619
Phase 2 Subphase 1 Iteration 200 Progress: 8%
theta 11.1925  8.4968 -2.2408  2.1720  0.6770 -0.1028 -0.1497  1.4084  2.2692
0.1253  0.0444  6.2007  0.0931
ac -0.119  0.200 -0.260 -0.294 -0.299 -0.255 -0.232  0.642  0.107 -0.295  0.274
```

```
0.450 -0.317
theta 10.5137  8.4097 -2.2157  2.2013  0.5062 -0.1032 -0.1769  1.5316  2.1799
0.1728  0.0673  7.0463  0.0793
ac -0.129  0.160 -0.257 -0.289 -0.294 -0.253 -0.234  0.621  0.101 -0.294  0.272
0.425 -0.312
theta: 10.5137  8.4097 -2.2157  2.2013  0.5062 -0.1032 -0.1769  1.5316  2.1799
0.1728  0.0673  7.0463  0.0793

Start phase 2.2
Phase 2 Subphase 2 Iteration 1 Progress: 8%
Phase 2 Subphase 2 Iteration 2 Progress: 8%
Phase 2 Subphase 2 Iteration 3 Progress: 8%
Phase 2 Subphase 2 Iteration 4 Progress: 8%
Phase 2 Subphase 2 Iteration 5 Progress: 8%
Phase 2 Subphase 2 Iteration 6 Progress: 8%
Phase 2 Subphase 2 Iteration 7 Progress: 8%
Phase 2 Subphase 2 Iteration 8 Progress: 8%
Phase 2 Subphase 2 Iteration 9 Progress: 8%
Phase 2 Subphase 2 Iteration 10 Progress: 8%
Phase 2 Subphase 2 Iteration 200 Progress: 10%
theta 10.8636  8.6627 -2.2893  2.1342  0.5588 -0.0850 -0.1548  1.5731  2.1684
0.1919  0.0619  6.9994  0.0705
ac -0.4165  0.0306 -0.7772 -0.7765 -0.7333 -0.7520 -0.6163 -0.0247 -0.1700
-0.5041 -0.2356 -0.2154 -0.5689
theta 10.7110  8.7601 -2.2738  2.1345  0.4862 -0.0841 -0.1619  1.5452  2.1748
0.1840  0.0730  7.3727  0.0724
ac -0.3774  0.0237 -0.7785 -0.7810 -0.7362 -0.7532 -0.6239 -0.0636 -0.1354
-0.5074 -0.2452 -0.1993 -0.5670
theta: 10.7110  8.7601 -2.2738  2.1345  0.4862 -0.0841 -0.1619  1.5452  2.1748
0.1840  0.0730  7.3727  0.0724

Start phase 2.3
Phase 2 Subphase 3 Iteration 1 Progress: 10%
Phase 2 Subphase 3 Iteration 2 Progress: 10%
Phase 2 Subphase 3 Iteration 3 Progress: 10%
Phase 2 Subphase 3 Iteration 4 Progress: 10%
Phase 2 Subphase 3 Iteration 5 Progress: 10%
Phase 2 Subphase 3 Iteration 6 Progress: 10%
Phase 2 Subphase 3 Iteration 7 Progress: 10%
Phase 2 Subphase 3 Iteration 8 Progress: 10%
Phase 2 Subphase 3 Iteration 9 Progress: 10%
Phase 2 Subphase 3 Iteration 10 Progress: 10%
Phase 2 Subphase 3 Iteration 1 Progress: 10%
Phase 2 Subphase 3 Iteration 2 Progress: 10%
Phase 2 Subphase 3 Iteration 3 Progress: 10%
Phase 2 Subphase 3 Iteration 4 Progress: 10%
Phase 2 Subphase 3 Iteration 5 Progress: 10%
Phase 2 Subphase 3 Iteration 6 Progress: 10%
```

```
Phase 2 Subphase 3 Iteration 7 Progress: 10%
Phase 2 Subphase 3 Iteration 8 Progress: 10%
Phase 2 Subphase 3 Iteration 9 Progress: 10%
Phase 2 Subphase 3 Iteration 10 Progress: 10%
Phase 2 Subphase 3 Iteration 1 Progress: 10%
Phase 2 Subphase 3 Iteration 2 Progress: 10%
Phase 2 Subphase 3 Iteration 3 Progress: 10%
Phase 2 Subphase 3 Iteration 4 Progress: 10%
Phase 2 Subphase 3 Iteration 5 Progress: 10%
Phase 2 Subphase 3 Iteration 6 Progress: 10%
Phase 2 Subphase 3 Iteration 7 Progress: 10%
Phase 2 Subphase 3 Iteration 8 Progress: 10%
Phase 2 Subphase 3 Iteration 9 Progress: 10%
Phase 2 Subphase 3 Iteration 10 Progress: 10%
Phase 2 Subphase 3 Iteration 1 Progress: 10%
Phase 2 Subphase 3 Iteration 2 Progress: 10%
Phase 2 Subphase 3 Iteration 3 Progress: 10%
Phase 2 Subphase 3 Iteration 4 Progress: 10%
Phase 2 Subphase 3 Iteration 5 Progress: 10%
Phase 2 Subphase 3 Iteration 6 Progress: 10%
Phase 2 Subphase 3 Iteration 7 Progress: 10%
Phase 2 Subphase 3 Iteration 8 Progress: 10%
Phase 2 Subphase 3 Iteration 9 Progress: 10%
Phase 2 Subphase 3 Iteration 10 Progress: 10%
Phase 2 Subphase 3 Iteration 200 Progress: 12%
theta 11.0048  8.9213 -2.2975  2.1435  0.4861 -0.0802 -0.1645  1.5824  2.1774
0.1928  0.0766  7.3594  0.0666
ac -0.04730  0.08017 -0.67224 -0.74477 -0.72800 -0.64541 -0.16958  0.00434
-0.03221 -0.02180  0.06525 -0.02856 -0.02465
theta 10.9770  8.9536 -2.2926  2.1180  0.4939 -0.0804 -0.1587  1.5698  2.1802
0.1920  0.0802  7.5745  0.0705
ac -0.02774  0.01329 -0.69291 -0.74311 -0.75388 -0.68922 -0.27917 -0.08363
-0.09532 -0.07220  0.01516  0.00157 -0.10450
theta: 10.9770  8.9536 -2.2926  2.1180  0.4939 -0.0804 -0.1587  1.5698  2.1802
0.1920  0.0802  7.5745  0.0705

Start phase 2.4
Phase 2 Subphase 4 Iteration 1 Progress: 12%
Phase 2 Subphase 4 Iteration 2 Progress: 12%
Phase 2 Subphase 4 Iteration 3 Progress: 12%
Phase 2 Subphase 4 Iteration 4 Progress: 12%
Phase 2 Subphase 4 Iteration 5 Progress: 12%
Phase 2 Subphase 4 Iteration 6 Progress: 12%
Phase 2 Subphase 4 Iteration 7 Progress: 12%
Phase 2 Subphase 4 Iteration 8 Progress: 12%
Phase 2 Subphase 4 Iteration 9 Progress: 12%
Phase 2 Subphase 4 Iteration 10 Progress: 12%
Phase 2 Subphase 4 Iteration 200 Progress: 14%
```

```
theta 10.9359  8.8728 -2.2898  2.0945  0.4943 -0.0800 -0.1558  1.5272  2.1709
0.1947  0.0840  7.6761  0.0719
ac -0.0756  0.0538 -0.2153 -0.4653 -0.4576 -0.2696 -0.2255  0.0870  0.0571
-0.0414  0.1586 -0.0910  0.0710
theta 10.9853  8.9495 -2.2933  2.1175  0.4952 -0.0802 -0.1550  1.5389  2.1666
0.2052  0.0781  7.5029  0.0681
ac -0.1051  0.0121 -0.2569 -0.3907 -0.3902 -0.2764 -0.1468  0.0487  0.0464
0.0554  0.1230 -0.0257  0.1165
theta: 10.9853  8.9495 -2.2933  2.1175  0.4952 -0.0802 -0.1550  1.5389  2.1666
0.2052  0.0781  7.5029  0.0681

Start phase 3
Phase 3 Iteration 1 Progress 15%
Phase 3 Iteration 401 Progress 19%
Phase 3 Iteration 801 Progress 22%
Phase 3 Iteration 1201 Progress 25%
Phase 3 Iteration 1601 Progress 29%
Phase 3 Iteration 2001 Progress 32%
Phase 3 Iteration 2401 Progress 36%
Phase 3 Iteration 2801 Progress 39%
Phase 3 Iteration 3201 Progress 42%
Phase 3 Iteration 3601 Progress 46%
Phase 3 Iteration 4001 Progress 49%
Phase 3 Iteration 4401 Progress 53%
Phase 3 Iteration 4801 Progress 56%
Phase 3 Iteration 5201 Progress 59%
Phase 3 Iteration 5601 Progress 63%
Phase 3 Iteration 6001 Progress 66%
Phase 3 Iteration 6401 Progress 70%
Phase 3 Iteration 6801 Progress 73%
Phase 3 Iteration 7201 Progress 76%
Phase 3 Iteration 7601 Progress 80%
Phase 3 Iteration 8001 Progress 83%
Phase 3 Iteration 8401 Progress 86%
Phase 3 Iteration 8801 Progress 90%
Phase 3 Iteration 9201 Progress 93%
Phase 3 Iteration 9601 Progress 97%

Estimates, standard errors and convergence t-ratios

                                              Estimate   Standard   Convergence
                                                           Error      t-ratio
Network Dynamics
   1. rate constant friendship rate (period 1) 10.9853  ( 1.0148   )   -0.0330
   2. rate constant friendship rate (period 2)  8.9495  ( 0.8403   )    0.0109
   3. eval outdegree (density)                 -2.2933  ( 0.0871   )   -0.0249
   4. eval reciprocity                          2.1175  ( 0.0901   )   -0.0064
   5. eval transitive triplets                  0.4952  ( 0.0317   )   -0.0191
```

```
   6. eval indegree - popularity              -0.0802  ( 0.0193   )   -0.0305
   7. eval log_dist                           -0.1550  ( 0.0446   )    0.0040

Behavior Dynamics
   8. rate rate alco_beh (period 1)            1.5389  ( 0.2482   )   -0.0281
   9. rate rate alco_beh (period 2)            2.1666  ( 0.3599   )   -0.0100
  10. eval alco_beh linear shape               0.2052  ( 0.2575   )    0.0202
  11. eval alco_beh quadratic shape            0.0781  ( 0.0820   )   -0.0012
  12. eval alco_beh average similarity         7.5029  ( 2.3707   )   -0.0119
  13. eval alco_beh indegree                   0.0681  ( 0.0691   )   -0.0024

Overall maximum convergence ratio:    0.0917


Total of 11314 iteration steps.


No X11 device available, forcing use of batch mode



Start phase 0
theta: 10.9853  8.9495 -2.2933  2.1175  0.4952 -0.0802 -0.1550  1.5389  2.1666
0.2052  0.0781  7.5029  0.0681

Start phase 2.1
Phase 2 Subphase 1 Iteration 1 Progress: 6%
Phase 2 Subphase 1 Iteration 2 Progress: 6%
theta 10.7817  9.1891 -2.2852  2.0956  0.4961 -0.0809 -0.1623  1.4322  2.1726
0.2030  0.0873  7.5525  0.0735
ac -2.53847 -0.78459 -0.00752 -1.96624  0.89520  0.41764  6.16285 -7.44850
-1.26461  0.20187  1.26064  0.68654  0.65685
Phase 2 Subphase 1 Iteration 3 Progress: 6%
Phase 2 Subphase 1 Iteration 4 Progress: 6%
theta 11.1853  8.8391 -2.2993  2.1230  0.4975 -0.0811 -0.1645  1.3692  2.2402
0.1988  0.0964  7.6673  0.0806
ac  1.0326 -0.7169  0.0425 -1.5685 -1.4242  0.4489  1.2039 -0.6146 -1.4975
0.1756  1.0552  0.7307  0.3205
Phase 2 Subphase 1 Iteration 5 Progress: 6%
Phase 2 Subphase 1 Iteration 6 Progress: 6%
theta 11.2366  8.9617 -2.2947  2.1089  0.5025 -0.0825 -0.1617  1.4614  2.2081
0.1814  0.1033  8.2931  0.0819
ac  1.27516 -0.71295  0.00492 -1.52726 -2.42442  0.25638  0.64127  0.26724
-1.35925  0.18781  0.94643  0.51233  0.37610
Phase 2 Subphase 1 Iteration 7 Progress: 6%
Phase 2 Subphase 1 Iteration 8 Progress: 6%
theta 10.9810  9.0500 -2.2829  2.1096  0.4984 -0.0842 -0.1622  1.6375  2.2221
0.1612  0.0885  8.0044  0.0772
ac  1.234 -0.723  0.264 -0.820 -1.499  0.269  0.688  0.271 -1.361  0.190  0.387
```

```
0.725  0.374
Phase 2 Subphase 1 Iteration 9 Progress: 6%
Phase 2 Subphase 1 Iteration 10 Progress: 6%
theta 10.6902  9.1680 -2.2758  2.0825  0.5063 -0.0827 -0.1553  1.6433  2.1762
0.1700  0.0941  7.8749  0.0713
ac  1.2425 -0.6839  0.2513 -0.2754  0.1051  0.3244 -0.0527  0.5115 -1.2446
-0.0493  0.3825  0.6748 -0.0418
theta 11.0125  8.9342 -2.2896  2.1069  0.4967 -0.0803 -0.1565  1.5426  2.1804
0.1785  0.0810  7.6013  0.0749
ac -0.1484 -0.2366 -0.0382 -0.1399 -0.0646 -0.1093 -0.0857 -0.2985 -0.1950
-0.0918 -0.0946 -0.0111 -0.0396
theta: 11.0125  8.9342 -2.2896  2.1069  0.4967 -0.0803 -0.1565  1.5426  2.1804
0.1785  0.0810  7.6013  0.0749

Start phase 2.2
Phase 2 Subphase 2 Iteration 1 Progress: 8%
Phase 2 Subphase 2 Iteration 2 Progress: 8%
Phase 2 Subphase 2 Iteration 3 Progress: 8%
Phase 2 Subphase 2 Iteration 4 Progress: 8%
Phase 2 Subphase 2 Iteration 5 Progress: 8%
Phase 2 Subphase 2 Iteration 6 Progress: 8%
Phase 2 Subphase 2 Iteration 7 Progress: 8%
Phase 2 Subphase 2 Iteration 8 Progress: 8%
Phase 2 Subphase 2 Iteration 9 Progress: 8%
Phase 2 Subphase 2 Iteration 10 Progress: 8%
theta 11.0492  8.8634 -2.2992  2.1119  0.4958 -0.0780 -0.1536  1.5471  2.1517
0.1964  0.0803  7.5904  0.0708
ac -0.16207 -0.08705 -0.08195 -0.16858 -0.16487 -0.18438 -0.00501 -0.18711
-0.22133 -0.16211 -0.03641 -0.01885 -0.16951
theta: 11.0492  8.8634 -2.2992  2.1119  0.4958 -0.0780 -0.1536  1.5471  2.1517
0.1964  0.0803  7.5904  0.0708

Start phase 2.3
Phase 2 Subphase 3 Iteration 1 Progress: 10%
Phase 2 Subphase 3 Iteration 2 Progress: 10%
Phase 2 Subphase 3 Iteration 3 Progress: 10%
Phase 2 Subphase 3 Iteration 4 Progress: 10%
Phase 2 Subphase 3 Iteration 5 Progress: 10%
Phase 2 Subphase 3 Iteration 6 Progress: 10%
Phase 2 Subphase 3 Iteration 7 Progress: 10%
Phase 2 Subphase 3 Iteration 8 Progress: 10%
Phase 2 Subphase 3 Iteration 9 Progress: 10%
Phase 2 Subphase 3 Iteration 10 Progress: 10%
Phase 2 Subphase 3 Iteration 200 Progress: 11%
theta 11.1222  8.9164 -2.2802  2.1256  0.4926 -0.0824 -0.1480  1.5462  2.1629
0.2048  0.0856  7.7915  0.0719
ac  0.134347 -0.035759  0.037426  0.000415  0.083470  0.051789  0.247351
-0.075096 -0.020535 -0.166676 -0.051235 -0.062490 -0.132404
```

```
theta 11.0073  8.9162 -2.2907  2.1175  0.4962 -0.0812 -0.1561  1.5525  2.1922
0.1953  0.0782  7.5224  0.0685
ac -0.02775 -0.05523  0.02993  0.00316  0.05712  0.04679  0.11493 -0.07085
-0.05095 -0.09524 -0.06402 -0.03754 -0.14121
theta: 11.0073  8.9162 -2.2907  2.1175  0.4962 -0.0812 -0.1561  1.5525  2.1922
0.1953  0.0782  7.5224  0.0685

Start phase 2.4
Phase 2 Subphase 4 Iteration 1 Progress: 12%
Phase 2 Subphase 4 Iteration 2 Progress: 12%
Phase 2 Subphase 4 Iteration 3 Progress: 12%
Phase 2 Subphase 4 Iteration 4 Progress: 12%
Phase 2 Subphase 4 Iteration 5 Progress: 12%
Phase 2 Subphase 4 Iteration 6 Progress: 12%
Phase 2 Subphase 4 Iteration 7 Progress: 12%
Phase 2 Subphase 4 Iteration 8 Progress: 12%
Phase 2 Subphase 4 Iteration 9 Progress: 12%
Phase 2 Subphase 4 Iteration 10 Progress: 12%
Phase 2 Subphase 4 Iteration 200 Progress: 14%
theta 11.0431  8.9438 -2.2989  2.1171  0.4948 -0.0784 -0.1563  1.5596  2.1300
0.2091  0.0825  7.4208  0.0642
ac  0.0480 -0.2525 -0.0374  0.0525  0.0219 -0.0131 -0.2360  0.0780  0.0577
0.0173 -0.0463 -0.1213  0.1189
theta 10.9677  8.9598 -2.2954  2.1135  0.4946 -0.0791 -0.1543  1.5488  2.1623
0.2053  0.0832  7.6786  0.0687
ac  0.061795 -0.118758  0.038671 -0.024058  0.080748  0.053534 -0.124280
-0.054690  0.012224  0.010670 -0.000407 -0.011317  0.078508
theta: 10.9677  8.9598 -2.2954  2.1135  0.4946 -0.0791 -0.1543  1.5488  2.1623
0.2053  0.0832  7.6786  0.0687

Start phase 3
Phase 3 Iteration 1 Progress 15%
Phase 3 Iteration 401 Progress 19%
Phase 3 Iteration 801 Progress 22%
Phase 3 Iteration 1201 Progress 25%
Phase 3 Iteration 1601 Progress 29%
Phase 3 Iteration 2001 Progress 32%
Phase 3 Iteration 2401 Progress 36%
Phase 3 Iteration 2801 Progress 39%
Phase 3 Iteration 3201 Progress 42%
Phase 3 Iteration 3601 Progress 46%
Phase 3 Iteration 4001 Progress 49%
Phase 3 Iteration 4401 Progress 53%
Phase 3 Iteration 4801 Progress 56%
Phase 3 Iteration 5201 Progress 59%
Phase 3 Iteration 5601 Progress 63%
Phase 3 Iteration 6001 Progress 66%
Phase 3 Iteration 6401 Progress 70%
```

```
Phase 3 Iteration 6801 Progress 73%
Phase 3 Iteration 7201 Progress 76%
Phase 3 Iteration 7601 Progress 80%
Phase 3 Iteration 8001 Progress 83%
Phase 3 Iteration 8401 Progress 86%
Phase 3 Iteration 8801 Progress 90%
Phase 3 Iteration 9201 Progress 93%
Phase 3 Iteration 9601 Progress 97%

No X11 device available, forcing use of batch mode



Start phase 0
theta:  7.454  6.829 -1.613  0.000  0.000  0.000  0.000  0.910  1.093  0.347
0.000  0.000  0.000

Start phase 1
Phase 1 Iteration 1 Progress: 0%
Phase 1 Iteration 3 Progress: 0%
Phase 1 Iteration 5 Progress: 0%
Phase 1 Iteration 15 Progress: 0%
Phase 1 Iteration 25 Progress: 0%
Phase 1 Iteration 35 Progress: 0%
Phase 1 Iteration 45 Progress: 0%
theta:  8.4542  7.4748 -1.7195  0.2385  0.2345 -0.0170 -0.0695  0.5233  1.0764
0.3218  0.0701  0.4488  0.0114

Start phase 2.1
Phase 2 Subphase 1 Iteration 1 Progress: 6%
Phase 2 Subphase 1 Iteration 2 Progress: 6%
theta  8.6451  7.7372 -1.7895  0.4264  0.3775 -0.0266 -0.0916  0.4363  1.1025
0.3067  0.0768  0.5325  0.0114
ac 1.147 1.266 0.862 1.334 1.677 0.753 2.002 0.927 0.888 0.895 1.061 0.877 1.009
Phase 2 Subphase 1 Iteration 3 Progress: 6%
Phase 2 Subphase 1 Iteration 4 Progress: 6%
theta  8.65739  8.23902 -1.98396  1.08997  0.53202 -0.04928 -0.12431  0.55298
1.20376  0.30078  0.04650  1.65835  0.00112
ac 1.239 1.430 1.187 1.313 1.924 1.278 2.071 1.032 0.799 1.014 1.231 0.962 0.958
Phase 2 Subphase 1 Iteration 5 Progress: 6%
Phase 2 Subphase 1 Iteration 6 Progress: 6%
theta  8.57576  8.69110 -2.20520  1.90790  0.47037 -0.06467 -0.14611  1.03496
1.51660  0.29780 -0.00582  4.70880 -0.00346
ac 1.218 1.459 1.217 1.317 1.670 1.310 2.045 1.014 0.882 1.066 1.140 0.995 0.990
Phase 2 Subphase 1 Iteration 7 Progress: 6%
Phase 2 Subphase 1 Iteration 8 Progress: 6%
theta  8.72466  8.83805 -2.32733  2.28352  0.43146 -0.07330 -0.15674  1.37914
1.74865  0.31476 -0.01369  6.73508  0.00303
ac 1.201 1.388 1.212 1.337 0.717 1.297 1.884 1.012 0.919 1.072 1.100 0.981 0.982
```

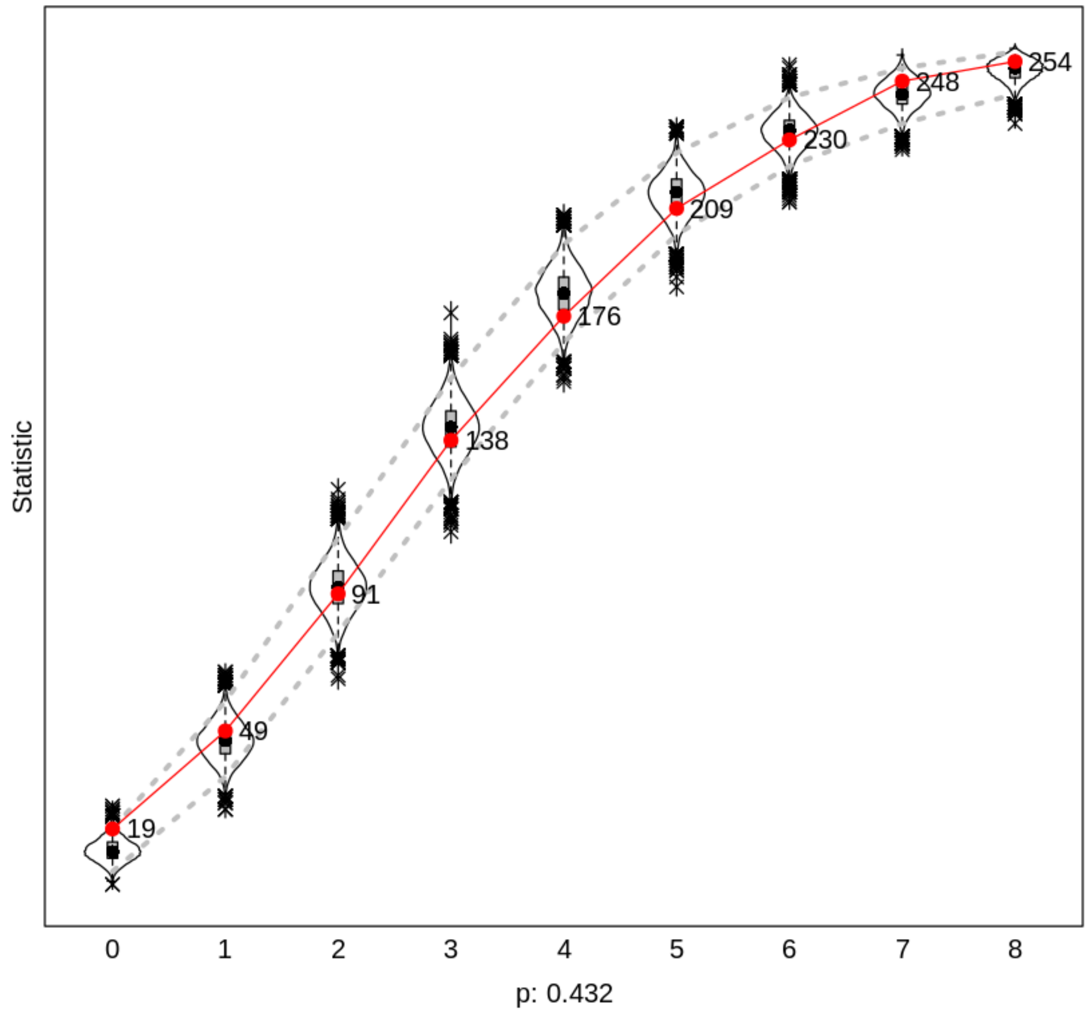
```
Phase 2 Subphase 1 Iteration 9 Progress: 6%
Phase 2 Subphase 1 Iteration 10 Progress: 6%
theta  8.96515  8.97531 -2.39760  2.45436  0.41098 -0.08068 -0.16823  1.62289
1.91085  0.34890 -0.00811  8.02617  0.01932
ac  1.084  1.373  1.137  1.306 -0.316  1.172  0.874  1.007  0.924  1.066  1.079
0.985  1.001
Phase 2 Subphase 1 Iteration 1 Progress: 6%
Phase 2 Subphase 1 Iteration 2 Progress: 6%
Phase 2 Subphase 1 Iteration 3 Progress: 6%
Phase 2 Subphase 1 Iteration 4 Progress: 6%
Phase 2 Subphase 1 Iteration 5 Progress: 6%
Phase 2 Subphase 1 Iteration 6 Progress: 6%
Phase 2 Subphase 1 Iteration 7 Progress: 6%
Phase 2 Subphase 1 Iteration 8 Progress: 6%
Phase 2 Subphase 1 Iteration 9 Progress: 6%
Phase 2 Subphase 1 Iteration 10 Progress: 6%
theta 10.4311  8.7246 -2.2732  2.1397  0.4964 -0.0854 -0.1665  1.4989  2.1847
0.2019  0.0822  7.7427  0.0675
ac -0.2507 -0.4856 -1.3343 -1.3855 -1.4176 -1.3508 -1.2124 -0.0773 -0.1062
-0.1309 -0.4163 -0.0031 -0.0676
theta: 10.4311  8.7246 -2.2732  2.1397  0.4964 -0.0854 -0.1665  1.4989  2.1847
0.2019  0.0822  7.7427  0.0675

Start phase 2.2
Phase 2 Subphase 2 Iteration 1 Progress: 8%
Phase 2 Subphase 2 Iteration 2 Progress: 8%
Phase 2 Subphase 2 Iteration 3 Progress: 8%
Phase 2 Subphase 2 Iteration 4 Progress: 8%
Phase 2 Subphase 2 Iteration 5 Progress: 8%
Phase 2 Subphase 2 Iteration 6 Progress: 8%
Phase 2 Subphase 2 Iteration 7 Progress: 8%
Phase 2 Subphase 2 Iteration 8 Progress: 8%
Phase 2 Subphase 2 Iteration 9 Progress: 8%
Phase 2 Subphase 2 Iteration 10 Progress: 8%
Phase 2 Subphase 2 Iteration 200 Progress: 9%
theta 10.9406  8.8122 -2.2801  2.1249  0.5064 -0.0845 -0.1497  1.5949  2.1791
0.2058  0.0562  7.3698  0.0636
ac  0.1061 -0.1229 -0.6635 -0.7527 -0.7551 -0.6619 -0.3317  0.0858  0.1132
-0.1966 -0.0058 -0.0727 -0.0506
theta 10.9557  8.9552 -2.2933  2.1213  0.4959 -0.0808 -0.1567  1.5526  2.1895
0.1975  0.0782  7.4687  0.0682
ac -0.00293 -0.10739 -0.64842 -0.71492 -0.72685 -0.64190 -0.30781  0.06792
0.15404 -0.14856  0.01782 -0.01087 -0.07616
theta: 10.9557  8.9552 -2.2933  2.1213  0.4959 -0.0808 -0.1567  1.5526  2.1895
0.1975  0.0782  7.4687  0.0682

Start phase 2.3
Phase 2 Subphase 3 Iteration 1 Progress: 10%
```

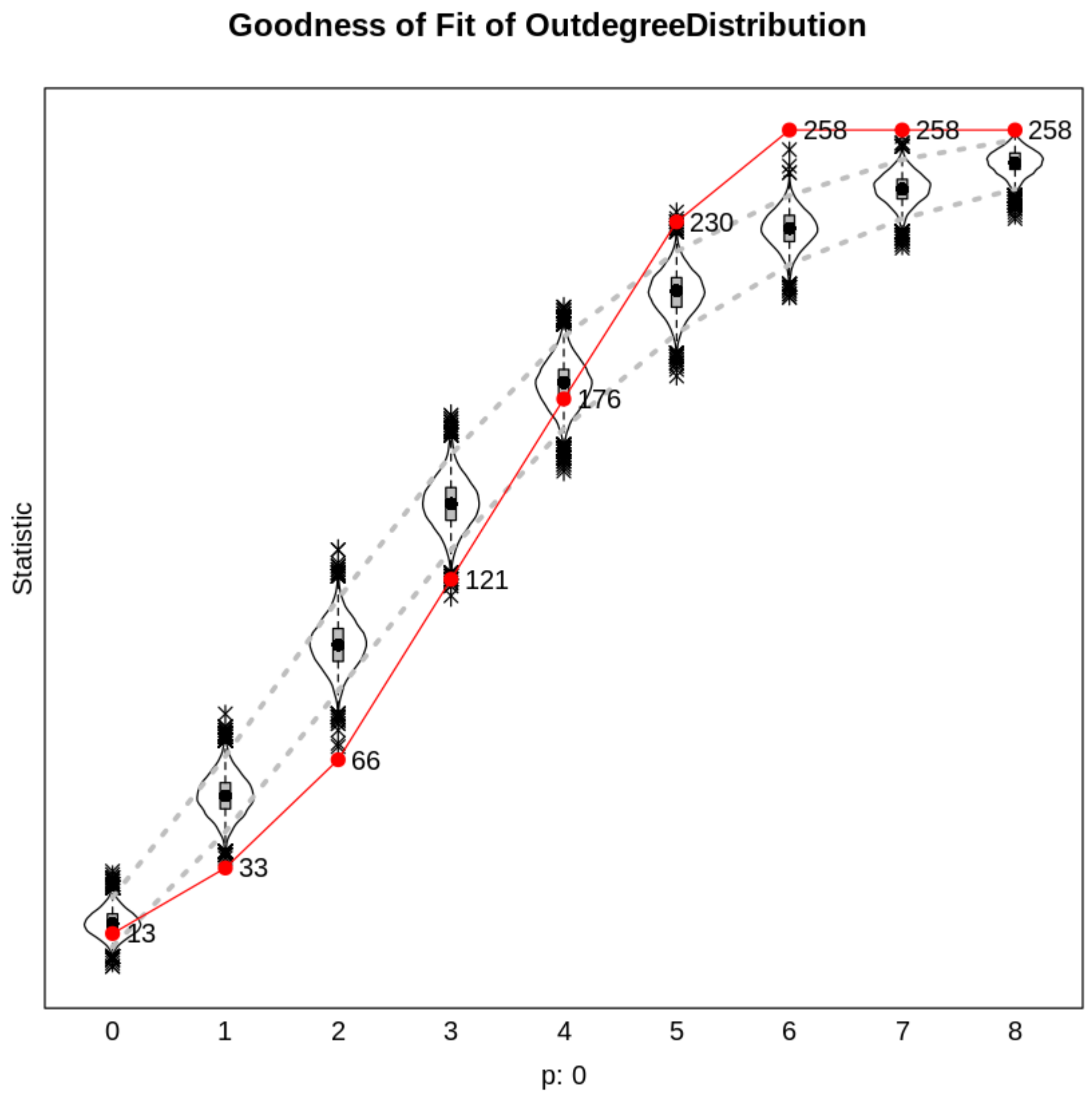
```
Phase 2 Subphase 3 Iteration 2 Progress: 10%
Phase 2 Subphase 3 Iteration 3 Progress: 10%
Phase 2 Subphase 3 Iteration 4 Progress: 10%
Phase 2 Subphase 3 Iteration 5 Progress: 10%
Phase 2 Subphase 3 Iteration 6 Progress: 10%
Phase 2 Subphase 3 Iteration 7 Progress: 10%
Phase 2 Subphase 3 Iteration 8 Progress: 10%
Phase 2 Subphase 3 Iteration 9 Progress: 10%
Phase 2 Subphase 3 Iteration 10 Progress: 10%
Phase 2 Subphase 3 Iteration 200 Progress: 12%
theta 10.9554  9.0562 -2.2816  2.1293  0.4810 -0.0835 -0.1536  1.5933  2.1451
0.1833  0.0840  7.7375  0.0780
ac -0.0819 -0.1574 -0.3161 -0.4493 -0.4063 -0.2797  0.0366  0.0455  0.1522
0.1092 -0.1426 -0.0544  0.0398
theta 11.0257  8.9950 -2.2838  2.1148  0.4972 -0.0823 -0.1544  1.5609  2.1770
0.1904  0.0826  7.6016  0.0715
ac -0.1033 -0.1112 -0.3083 -0.4340 -0.4664 -0.3123 -0.0741  0.0962  0.0191
0.1610 -0.0409 -0.0256  0.0367
theta: 11.0257  8.9950 -2.2838  2.1148  0.4972 -0.0823 -0.1544  1.5609  2.1770
0.1904  0.0826  7.6016  0.0715

Start phase 2.4
Phase 2 Subphase 4 Iteration 1 Progress: 13%
Phase 2 Subphase 4 Iteration 2 Progress: 13%
Phase 2 Subphase 4 Iteration 3 Progress: 13%
Phase 2 Subphase 4 Iteration 4 Progress: 13%
Phase 2 Subphase 4 Iteration 5 Progress: 13%
Phase 2 Subphase 4 Iteration 6 Progress: 13%
Phase 2 Subphase 4 Iteration 7 Progress: 13%
Phase 2 Subphase 4 Iteration 8 Progress: 13%
Phase 2 Subphase 4 Iteration 9 Progress: 13%
Phase 2 Subphase 4 Iteration 10 Progress: 13%
Phase 2 Subphase 4 Iteration 200 Progress: 14%
theta 10.8749  8.9254 -2.3034  2.1108  0.4880 -0.0773 -0.1532  1.5598  2.1608
0.1616  0.0862  7.8050  0.0803
ac -0.1323  0.0719 -0.2312 -0.2728 -0.3589 -0.3399 -0.2867  0.0819  0.1422
0.1128 -0.0365 -0.0956  0.0932
Phase 2 Subphase 4 Iteration 400 Progress: 16%
theta 11.0488  8.8961 -2.2957  2.1272  0.4986 -0.0821 -0.1597  1.5471  2.1683
0.1995  0.0695  7.3950  0.0701
ac -0.0432  0.0852 -0.2477 -0.3077 -0.2678 -0.2884 -0.2434 -0.0313  0.0347
0.0125  0.0378  0.0426 -0.0346
theta 10.9626  8.9202 -2.2942  2.1165  0.4960 -0.0800 -0.1557  1.5446  2.1730
0.1797  0.0777  7.5314  0.0743
ac -0.04186  0.07976 -0.21331 -0.29243 -0.27164 -0.23948 -0.19281 -0.05026
0.02776  0.00151 -0.00710 -0.00590 -0.03323
theta: 10.9626  8.9202 -2.2942  2.1165  0.4960 -0.0800 -0.1557  1.5446  2.1730
0.1797  0.0777  7.5314  0.0743
```

```
Start phase 3
Phase 3 Iteration 1 Progress 18%
Phase 3 Iteration 201 Progress 19%
Phase 3 Iteration 401 Progress 21%
Phase 3 Iteration 601 Progress 23%
Phase 3 Iteration 801 Progress 24%
Phase 3 Iteration 1001 Progress 26%
Phase 3 Iteration 1201 Progress 28%
Phase 3 Iteration 1401 Progress 29%
Phase 3 Iteration 1601 Progress 31%
Phase 3 Iteration 1801 Progress 32%
Phase 3 Iteration 2001 Progress 34%
Phase 3 Iteration 2201 Progress 36%
Phase 3 Iteration 2401 Progress 37%
Phase 3 Iteration 2601 Progress 39%
Phase 3 Iteration 2801 Progress 41%
Phase 3 Iteration 3001 Progress 42%
Phase 3 Iteration 3201 Progress 44%
Phase 3 Iteration 3401 Progress 46%
Phase 3 Iteration 3601 Progress 47%
Phase 3 Iteration 3801 Progress 49%
Phase 3 Iteration 4001 Progress 51%
Phase 3 Iteration 4201 Progress 52%
Phase 3 Iteration 4401 Progress 54%
Phase 3 Iteration 4601 Progress 56%
Phase 3 Iteration 4801 Progress 57%
Phase 3 Iteration 5001 Progress 59%
Phase 3 Iteration 5201 Progress 60%
Phase 3 Iteration 5401 Progress 62%
Phase 3 Iteration 5601 Progress 64%
Phase 3 Iteration 5801 Progress 65%
Phase 3 Iteration 6001 Progress 67%
Phase 3 Iteration 6201 Progress 69%
Phase 3 Iteration 6401 Progress 70%
Phase 3 Iteration 6601 Progress 72%
Phase 3 Iteration 6801 Progress 74%
Phase 3 Iteration 7001 Progress 75%
Phase 3 Iteration 7201 Progress 77%
Phase 3 Iteration 7401 Progress 79%
Phase 3 Iteration 7601 Progress 80%
Phase 3 Iteration 7801 Progress 82%
Phase 3 Iteration 8001 Progress 84%
Phase 3 Iteration 8201 Progress 85%
Phase 3 Iteration 8401 Progress 87%
Phase 3 Iteration 8601 Progress 88%
Phase 3 Iteration 8801 Progress 90%
Phase 3 Iteration 9001 Progress 92%
```

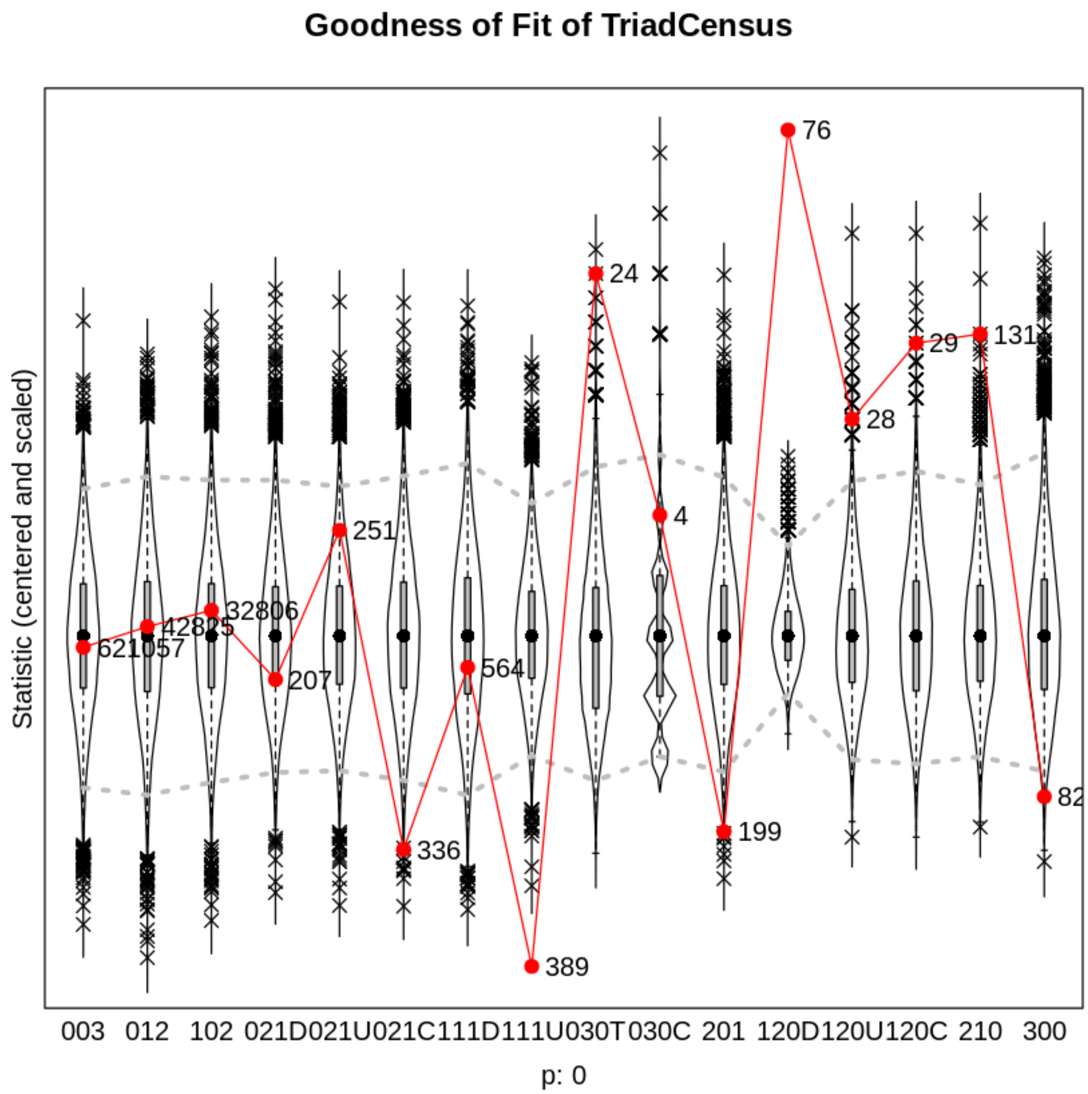
```
Phase 3 Iteration 9201 Progress 93%
Phase 3 Iteration 9401 Progress 95%
Phase 3 Iteration 9601 Progress 97%
Phase 3 Iteration 9801 Progress 98%
1 0.03522177 0.09226922

Estimates, standard errors and convergence t-ratios

                                              Estimate   Standard   Convergence
                                                           Error      t-ratio
Network Dynamics
   1. rate constant friendship rate (period 1) 10.9626  ( 1.0240   )   -0.0324
   2. rate constant friendship rate (period 2)  8.9202  ( 0.7701   )   -0.0100
   3. eval outdegree (density)                 -2.2942  ( 0.0892   )    0.0082
   4. eval reciprocity                          2.1165  ( 0.0920   )    0.0223
   5. eval transitive triplets                  0.4960  ( 0.0306   )    0.0091
   6. eval indegree - popularity               -0.0800  ( 0.0195   )    0.0116
   7. eval log_dist                            -0.1557  ( 0.0450   )   -0.0189

Behavior Dynamics
   8. rate rate alco_beh (period 1)             1.5446  ( 0.2502   )   -0.0098
   9. rate rate alco_beh (period 2)             2.1730  ( 0.3566   )    0.0107
  10. eval alco_beh linear shape                0.1797  ( 0.2570   )    0.0081
  11. eval alco_beh quadratic shape             0.0777  ( 0.0839   )   -0.0169
  12. eval alco_beh average similarity          7.5314  ( 2.3959   )   -0.0133
  13. eval alco_beh indegree                    0.0743  ( 0.0681   )    0.0352

Overall maximum convergence ratio:    0.0923


Total of 11421 iteration steps.
```

Goodness of Fit of IndegreeDistribution



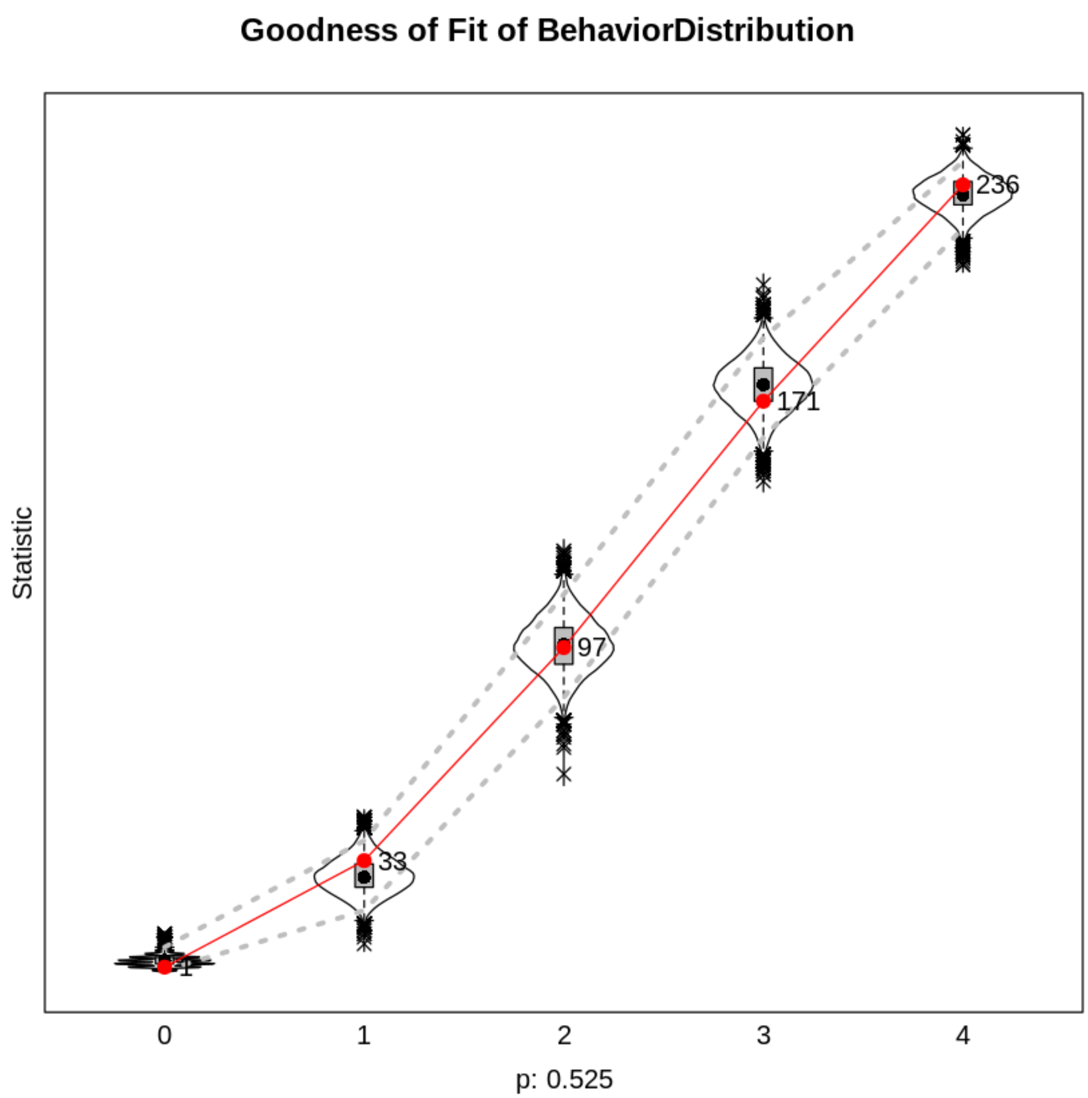



```
Note: some statistics are not plotted because their variance is 0.
This holds for the statistic: 5.
```



A matrix: 10×5 of type dbl

	0	1	2	3	4
max	11.000000	46.000000	126.000000	206.000000	251.000000
perc.upper	7.000000	39.000000	113.000000	190.000000	243.000000
mean	2.894700	28.307900	97.814100	175.758500	233.248000
median	3.000000	28.000000	98.000000	176.000000	233.000000
perc.lower	0.000000	18.000000	82.000000	160.000000	223.000000
min	0.000000	8.000000	59.000000	147.000000	212.000000
sd	1.750748	5.368014	7.993043	7.678989	5.212582
obs	1.000000	33.000000	97.000000	171.000000	236.000000
p>	0.776500	0.167900	0.507600	0.715800	0.271200
p>=	0.943300	0.215700	0.559000	0.759600	0.338000



[ ]:
```
siena.table(model1, type = "html", sig = TRUE)
```

Results for model1 written to model1.html .

[ ]:
```
siena.table(model1, type = "tex", sig = TRUE)
```

Results for model1 written to model1.tex .

[ ]:
```
model1
```

Estimates, standard errors and convergence t-ratios

```
                                      Estimate   Standard   Convergence
                                                   Error      t-ratio
```

```
Network Dynamics
   1. rate constant friendship rate (period 1) 10.9677  ( 1.0351   )   -0.0161
   2. rate constant friendship rate (period 2)  8.9598  ( 0.7702   )    0.0484
   3. eval outdegree (density)                 -2.2954  ( 0.0896   )   -0.0248
   4. eval reciprocity                          2.1135  ( 0.0912   )   -0.0392
   5. eval transitive triplets                  0.4946  ( 0.0316   )   -0.0304
   6. eval indegree - popularity               -0.0791  ( 0.0196   )   -0.0147
   7. eval log_dist                            -0.1543  ( 0.0442   )    0.0056

Behavior Dynamics
   8. rate rate alco_beh (period 1)             1.5488  ( 0.2541   )   -0.0094
   9. rate rate alco_beh (period 2)             2.1623  ( 0.3487   )   -0.0381
  10. eval alco_beh linear shape                0.2053  ( 0.2592   )   -0.0013
  11. eval alco_beh quadratic shape             0.0832  ( 0.0832   )   -0.0102
  12. eval alco_beh average similarity          7.6786  ( 2.4532   )    0.0499
  13. eval alco_beh indegree                    0.0687  ( 0.0692   )   -0.0268

Overall maximum convergence ratio:    0.1228


Total of 10962 iteration steps.
```

Comments: If we consider the indegree distribution or behavior distribution as auxiliary statistics, the model fits good (the red line lies in the 95% percent interval). But if we consider the outdegree distribution or triad census as auxiliary statistics, then the model fits not so good: sometimes red line is outside the 95% percent interval. But overall, it happens quite rarely, so we may assume that the model fits well enough.

(2.4) Are the hypothesis supported by the data? Argue for your answers.

Answer:

1. The indegree popularity. This statistics is significant under the level of 0.001. The sign of the estimated value is negative, which contradicts with the hypothesis H1.

2. The centered covariate main effect of logarithm of the distance between the houses of the pupils. This statistics is significant under the level of 0.001. The sign of the estimated value is negative, which supports the hypothesis H2.

3. The the indegree effect for behavioral evaluation function. This statistics is not significant even under the level of 0.1.

4. The average similarity effect. This statistics is significant under the level of 0.01. The sign of the estimated value is positive, which supports the hypothesis H4.

(2.5) Given the estimated model, do we have evidence for selection processes only, influece processes only, both selection and influence processes, or neither? Argue for your answer.

Answer: 1. The hypotheses H1 and H2 are examples of selection processes. H1 describes a scenario where students are actively choosing their friends based on a specific attribute (popularity). It highlights the selection mechanism where students are drawn to others with certain social traits.

H2 also reflects a selection process, as the proximity (living nearby) serves as a criterion or facilitator for choosing friendships. This is about selecting friends based on geographic or situational factors. Thus we see evidence of selection processes.

2. On the other hand, the hypotheses H3 and H4 are examples of influence processes. H3 is not statistically relevant, so we concentrate on H4. It describes an influence process because it focuses on how students' behaviors (alcohol consumption) are shaped by their social relationships (friends' behavior). It reflects the adjustment or conformity to peer norms. Thus we see evidence of influence processes.

(2.6) Discuss how the model could be improved (by adding new effects) so that geodesic distances and degree distributions might be better represented. Provide theoretical justification for the new effects you propose to add in the model specification. You are not required to re-run the model with the new proposed specification.

(2.7) Could you think of two other hypotheses concerning the dynamics of friendship and alcohol consumption dynamics that a researcher can test using SAOMs? State these hypotheses and how you would operationalize the corresponding effects in the evaluation function.

Answer:

1. The first hypothesis is that male students tend to consume more alcohol. To test this we can add the main covariate effect (effFrom) to the objective function.
2. The second hypothesis is that students of the same gender tend to be friends more than students of the oppostite genders. To test this we can add covariate-related similarity (simX) effect to the objective function.