Assignment 2 - Network Modeling

Group 5

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

2023-12-06

Task 1

(1)

- Out-degree:

$$s_{1i}(x) = \sum_j x_{ij}$$

- Reciprocity:

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

- Transitive reciprocated triplets effect:

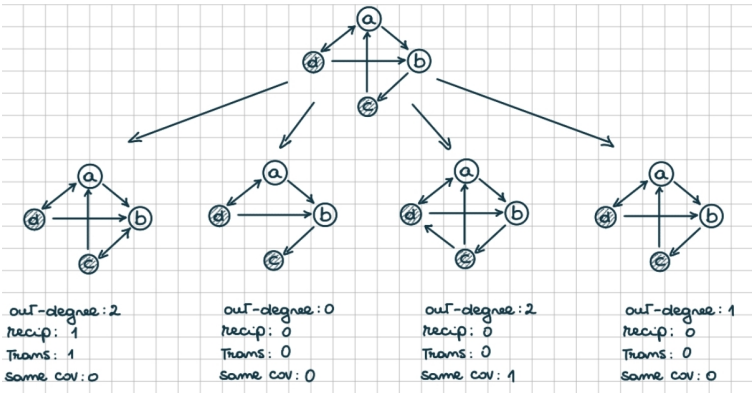
$$s_{3i}(x) = \sum_{j,h} x_{ij}x_{ji}x_{ih}x_{hj}$$

- Same covariate effect:

$$s_{4i}(x) = \sum_j x_{ij}I\{v_i = v_j\}$$

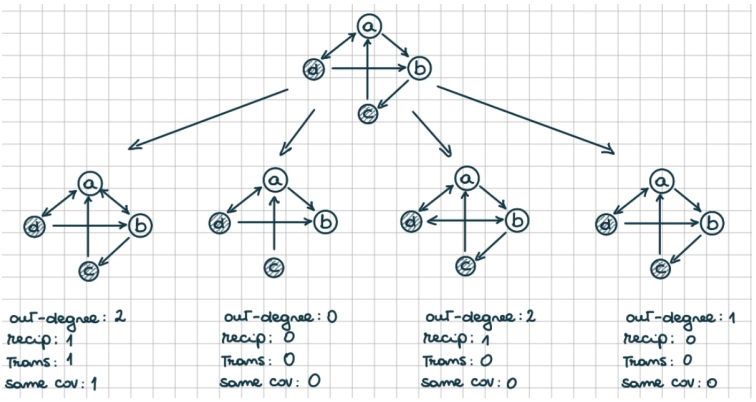
(2)

(i) Actor c can either add a tie to b, remove the tie to a, add a tie to d or keep the network unchanged.



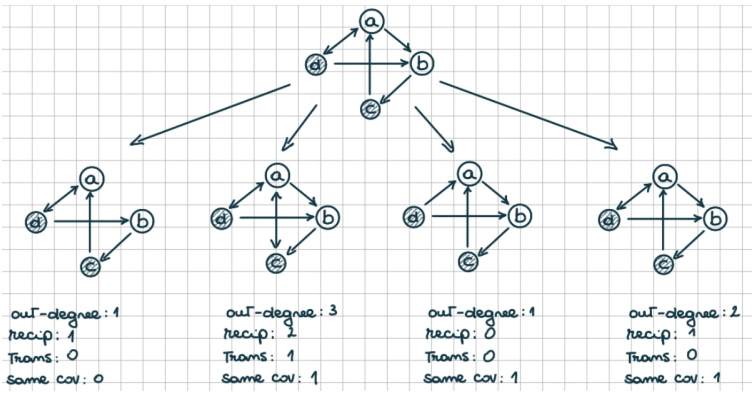
$$\frac{exp(-1.5\times2+2\times1+1\times1+1.5\times0)}{exp(-1.5\times2+2\times1+1\times1+1.5\times0)+exp(-1.5\times0+2\times0+1\times0+1.5\times0)+exp(-1.5\times2+2\times0+1\times0+1.5\times1)+exp(-1.5\times1+2\times0+1\times0+1.5\times0)} = 0.4087872$$

(ii) Actor b can either add a tie to a, remove the tie to c, add a tie to d or keep the network unchanged.



$$\frac{exp(-1.5\times2+2\times1+1\times1+1.5\times1)}{exp(-1.5\times2+2\times1+1\times1+1.5\times1)+exp(-1.5\times0+2\times0+1\times0+1.5\times0)+exp(-1.5\times2+2\times1+1\times0+1.5\times0)+exp(-1.5\times1+2\times0+1\times0+1.5\times0)} = 0.7380062$$

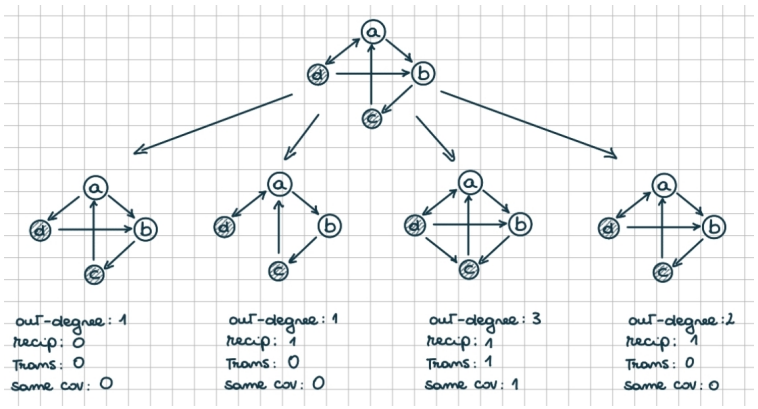
(iii) Actor a can either remove the tie to b, add a tie to c, remove the tie to d or keep the network unchanged.



$$\frac{exp(-1.5\times1+2\times1+1\times0+1.5\times0)}{exp(-1.5\times1+2\times1+1\times0+1.5\times0)+exp(-1.5\times3+2\times2+1\times1+1.5\times1)+exp(-1.5\times1+2\times0+1\times0+1.5\times1)+exp(-1.5\times2+2\times1+1\times0+1.5\times1)} = 0.1410791$$

(iv) Actor a can either remove the tie to a, remove the tie to b, add a tie to c or keep the network unchanged.

$$\frac{exp(-1.5\times2+2\times1+1\times0+1.5\times0)}{exp(-1.5\times1+2\times0+1\times0+1.5\times0)+exp(-1.5\times1+2\times1+1\times0+1.5\times0)+exp(-1.5\times3+2\times1+1\times1+1.5\times1)+exp(-1.5\times2+2\times1+1\times0+1.5\times0)} = 0.1135525$$



Task 2

(1)

```
# Task 2.1 ------------------------------------------------------------------
# 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 transTrip 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 lambda rate parameter
#' @param beta1 outdegree parameter
#' @param beta2 reciprocity parameter
#' @param beta3 transTrip parameter
#'
#' @return network at time t2
#'
#' @examples
#' netT1 <- matrix(c(
#'   0, 1, 0, 0, 0,
#'   0, 0, 0, 1, 0,
#'   0, 0, 0, 1, 1,
#'   1, 0, 1, 0, 0,
#'   0, 1, 1, 0, 1
#'   ),
#'   nrow = 5, ncol =  5, byrow = TRUE)
#' netT2 <- simulation(5, netT1, 4, -2, 0.5, 0.05)
simulation <- function(n, x1, lambda, beta1, beta2, beta3) {
  t <- 0 # time
  x <- x1
```

```
  while (t < 1) {
    dt <- rexp(1, n * lambda)
    i <- sample(1:n, size=1)

    delta_outdegree <- rep(0, n)
    delta_rec <- rep(0, n)
    delta_trans_trip <- rep(0, n)

    for (j in (1:n)) {
      if (j==i) next

      delta_outdegree[j] <- 1 - 2*x[i,j]
      delta_rec[j] <- (1 - 2*x[i,j])*x[j,i]
      delta_trans_trip[j] <- (1 - 2*x[i,j]) * x[i,] %*% (x[,j] + x[j,])
    }

    p <- exp(beta1 * delta_outdegree + beta2 * delta_rec + beta3 * delta_trans_trip) /
      sum(exp(beta1 * delta_outdegree + beta2 * delta_rec + beta3 * delta_trans_trip))
    j <- which.max(rmultinom(1, 1, prob = p))
    if (i != j) x[i,j] <- 1 - x[i,j]
    t <- t+dt
  }
  return(x)
}
```

Assuming the tie $i \to j$ is toggled, only one value in the adjacency matrix is going to change: $x[i,j]$ becomes $1 - x[i,j]$.

Change in out-degree: The out-degree of node i is the sum of the elements of the ith row of the adjacency matrix and the only term of the sum that changes is $x[i,j]$. It follows that the delta of the out-degree is $(1 - x[i,j]) - x[i,j] = 1 - 2x[i,j]$.

Change in reciprocated pairs: The number of reciprocated pairs is given by $\sum_{a<b} x[a,b] \cdot x[b,a]$: the only term that changes by toggling edge $i \to j$ corresponds to $\{a,b\} = \{i,j\}$. Therefore, the delta of the number of reciprocated pairs is $(1 - x[i,j]) \cdot x[j,i] - x[i,j] \cdot x[j,i] = (1 - 2x[i,j]) \cdot x[j,i]$.

Change in transitive triplets: The number of transitive reciprocated triplets is given by $\sum_{a,b,c} x[a,c] \cdot x[c,b] \cdot x[a,b]$. There are two type of terms that change by toggling edge $i \to j$: those of the form $x[i,j] \cdot x[j,h] \cdot x[i,h]$, i.e. the added/removed edge adds/removes a 2-path; those of the form $x[i,h] \cdot x[h,j] \cdot x[i,j]$, i.e. the added/removed edge closes/opens a 2-path. Therefore, the total delta is given by

$$\sum_h (1 - 2x[i,j]) \cdot x[j,h] \cdot x[i,h] + \sum_h x[i,h] \cdot x[h,j] \cdot (1 - 2x[i,j]) = (1 - 2x[i,j]) \sum_h x[i,h] \cdot (x[j,h] + x[h,j])$$

.

(2)

```
net1 <- as.matrix(read.csv('net1.csv', header=F))
net2 <- as.matrix(read.csv('net2.csv', header=F))

waves <- sienaDependent(array(c(net1, net2), dim=c(22, 22, 2)))
myData <- sienaDataCreate(waves)
myData
```

```
## Dependent variables:  waves
## Number of observations: 2
##
## Nodeset                  Actors
## Number of nodes              22
##
## Dependent variable waves
## Type               oneMode
## Observations       2
## Nodeset            Actors
## Densities          0.17 0.17
```

```
myeff <- getEffects(myData)
myeff <- includeEffects(myeff, transTrip)
```

```
##   effectName          include fix   test  initialValue parm
## 1 transitive triplets TRUE    FALSE FALSE          0   0
```

```
myAlgorithm <- sienaAlgorithmCreate(
  nsub = 2, n3 = 3000, seed = 2023
)
```

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

```
model0 <- siena07(myAlgorithm,
  data = myData, effects = myeff, returnDeps = TRUE,
  useCluster = TRUE, nbrNodes = 4
)
model0
```

```
## Estimates, standard errors and convergence t-ratios
##
##                              Estimate   Standard   Convergence
##                                           Error      t-ratio
##
## Rate parameters:
##   0       Rate parameter       4.1444   ( 0.6933   )
##
## Other parameters:
##   1. eval outdegree (density) -1.1067  ( 0.1940   )   0.0158
##   2. eval reciprocity          0.4817  ( 0.3105   )   0.0379
##   3. eval transitive triplets  0.0774  ( 0.0932   )   0.0330
##
## Overall maximum convergence ratio:    0.0514
##
##
## Total of 3122 iteration steps.
```

```
(lambda <- model0$rate)
```

```
## [1] 4.144419
```

```r
(beta <- model0$theta)
```

```
## [1] -1.10667068  0.48170982  0.07740283
```

(3)

```r
set.seed(2023)
nsim <- 1000
triadCensus <- matrix(NA, nrow=nsim, ncol=16) # 16 types of triads
for (i in 1:nsim) {
  x2sim <- simulation(nrow(net1), net1, lambda, beta[1], beta[2], beta[3])
  triadCensus[i,] <- triad.census(x2sim, mode = 'digraph')
}
```

(4)

```r
## i. standardized the simulated network stats. ----
##   Name the resulting object as triadCensusStd
stdev <- apply(triadCensus, 2, sd)
m <- apply(triadCensus, 2, mean)
triadCensusStd <- sapply(1:length(m), function(x) (triadCensus[,x] - m[x])/stdev[x])

## ii. variance-covariance matrix and its generalized inverse.         ----
Phatinv <- MASS::ginv(cov(triadCensusStd))

## iii. standardized the observed values of the triad census counts    ----
##  in the second observation using values from i.
net2std <- sapply(1:length(m), function(x) (triad.census(net2)[,x] - m[x])/stdev[x])
names(net2std) <- 1:16

## 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
}

triadCensusStd <- triadCensusStd[,stdev>0]
```

```r
mhd_sim <- apply(triadCensusStd, 1, function(x) mhd(x, Phatinv))
mhd_obs <- as.numeric(mhd(net2std, Phatinv))

## 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.
mean(mhd_sim >= mhd_obs)
```

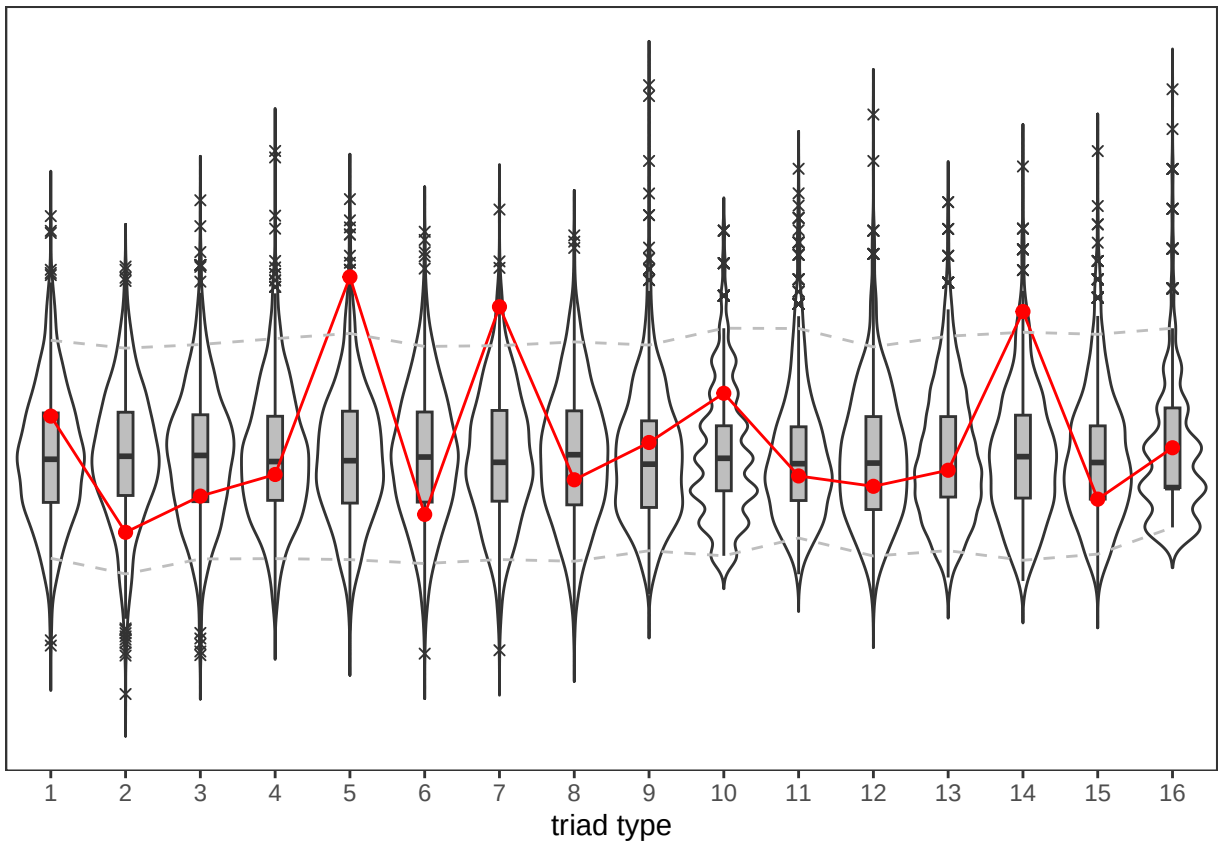
```
## [1] 0.004
```

```r
# 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
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(ggplot2)

# 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.
triadCensusObs <- # ---MISSING--- |>
  data.frame(t(net2std)) |>
  pivot_longer(
    everything(),
    names_to = "triad", names_pattern = "^X(.+)$",
    values_to = "nnodes"
  ) |>
  filter(triad %in% unique(triadCensusDf$triad))
```

```r
# 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"
  )


# 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")
```



The p-value computed in (4) suggests that the model has a bad fit, because it is below 0.05. Considering the plot, 3 observed values are outside the 95% confidence intervals.

Task 3

(1) (1.1)

```
f1 <- as.matrix(read.csv("f1.csv", header = FALSE))
f2 <- as.matrix(read.csv("f2.csv", header = FALSE))
f3 <- as.matrix(read.csv("f3.csv", header = FALSE))
attributes <- read.csv("demographic.csv", header = TRUE)
logdistance <- as.matrix(read.csv("logdistance.csv", header = FALSE))
alcohol <- as.matrix(read.csv('alcohol.csv', header = T))

friendship <- sienaDependent(
  array(c(f1, f2, f3), dim = c(129, 129, 3))
)

gender <- coCovar(attributes$gender)
age <- coCovar(attributes$age)
logdistance <- coDyadCovar(logdistance)
alcohol <- varCovar(alcohol)

mydata <- sienaDataCreate(friendship, gender, age, logdistance, alcohol)
print01Report(mydata, modelname='Glasgow')
```

```
# Tie changes between subsequent observations:
#  periods        0 =>  0   0 =>  1   1 =>  0   1 =>  1   Distance Jaccard   Missing
#   1 ==>   2     15827       237       240       208       477     0.304         0 (0%)
#   2 ==>   3     15839       228       209       236       437     0.351         0 (0%)
```

The Jaccard Index is above 0.3 in both periods, so the data contains enough information to investigate the evolution of the friendship network.

(1.2)

```
myeff <- getEffects(mydata)
myAlgorithm <- sienaAlgorithmCreate(
  projname = "friends_res",
  nsub = 4, n3 = 3000, seed = 2023
)
```

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

```
myeff <- includeEffects(myeff, transTrip)
```

```
##   effectName          include fix   test  initialValue parm
## 1 transitive triplets TRUE    FALSE FALSE          0   0
```

```
myeff <- includeEffects(myeff, inPop)
```

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

```
myeff <- includeEffects(myeff, egoX, altX, simX, interaction1='alcohol')
```

```
##   effectName         include fix   test  initialValue parm
## 1 alcohol alter      TRUE    FALSE FALSE          0   0
## 2 alcohol ego        TRUE    FALSE FALSE          0   0
## 3 alcohol similarity TRUE    FALSE FALSE          0   0
```

```
myeff <- includeEffects(myeff, X, interaction1='logdistance')
```

```
##   effectName  include fix   test  initialValue parm
## 1 logdistance TRUE    FALSE FALSE          0   0
```

(1.3)

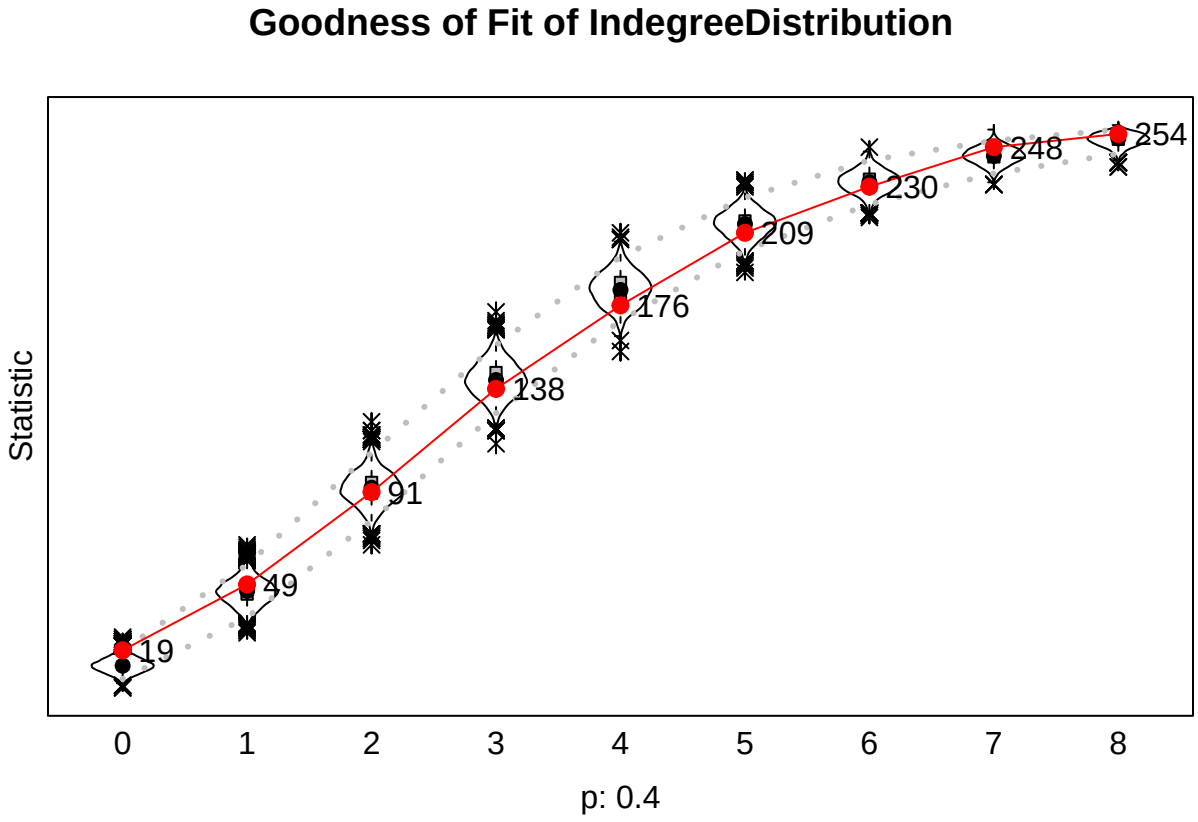
```
modelEv <- siena07(
  myAlgorithm,
  data = mydata, effects = myeff,
  returnDeps = TRUE,
  useCluster = TRUE, nbrNodes = 4
)
modelEv
```

```
## Estimates, standard errors and convergence t-ratios
##
##                                    Estimate   Standard   Convergence
##                                               Error      t-ratio
##
## Rate parameters:
##   0.1      Rate parameter period 1 10.8685  ( 1.0110   )
##   0.2      Rate parameter period 2  8.9120  ( 0.7996   )
##
## Other parameters:
##   1.  eval outdegree (density)     -2.3960  ( 0.0920   )    0.0017
##   2.  eval reciprocity              2.4325  ( 0.1075   )   -0.0164
##   3.  eval transitive triplets      0.7513  ( 0.0466   )   -0.0003
##   4.  eval 3-cycles                -0.5885  ( 0.0922   )   -0.0107
##   5.  eval indegree - popularity   -0.0875  ( 0.0204   )    0.0036
##   6.  eval logdistance             -0.1718  ( 0.0466   )   -0.0083
##   7.  eval alcohol alter            0.0321  ( 0.0307   )    0.0110
##   8.  eval alcohol ego             -0.0129  ( 0.0293   )   -0.0170
##   9.  eval alcohol similarity       0.6392  ( 0.1588   )    0.0575
##
## Overall maximum convergence ratio:    0.0838
##
##
## Total of 3758 iteration steps.
```

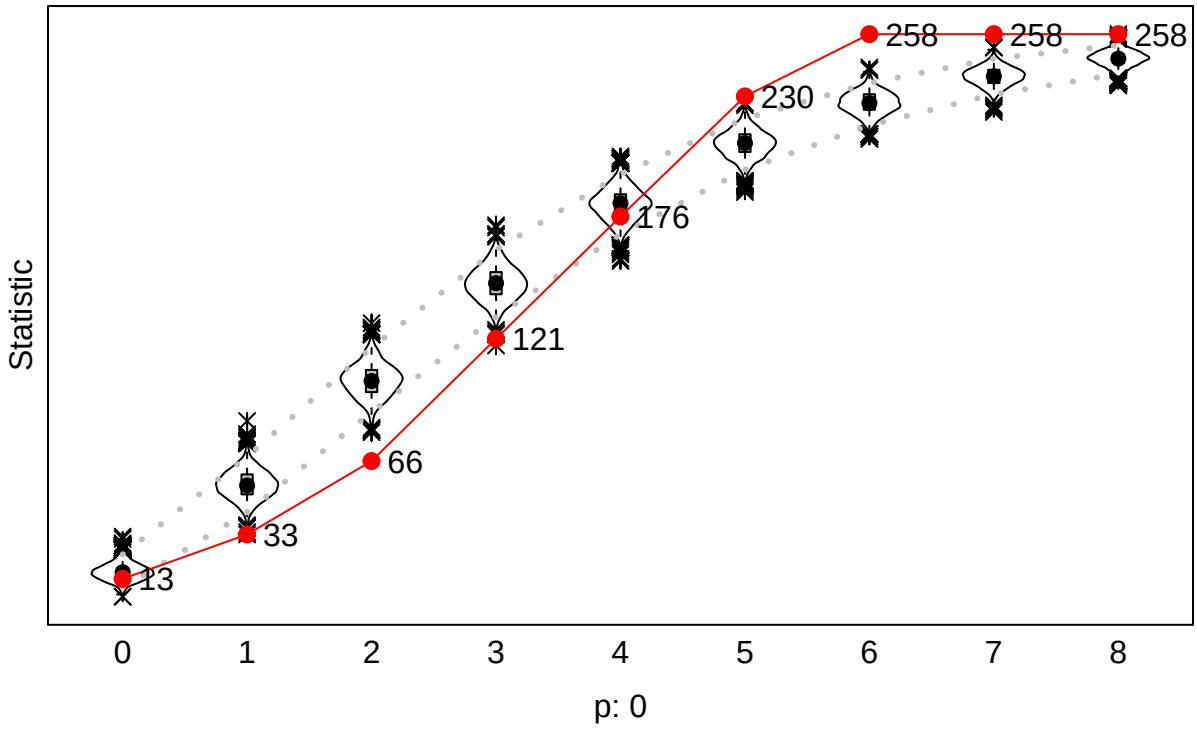
The algorithm converged, because the absolute value of the maximum t-ratio of all the parameters is below 0.1 and the overall maximum convergence ratio is below 0.2.

```r
# use parallel computation
cl <- makeCluster(4) # with 4 workers

# Indegree distribution
gofEvId <- sienaGOF(
  modelEv,
  verbose = FALSE,
  varName = "friendship", IndegreeDistribution,
  cluster = cl
)

# Outdegree distribution
gofEvOd <- sienaGOF(
  modelEv,
  verbose = FALSE,
  varName = "friendship", OutdegreeDistribution,
  cluster = cl
)

# Triad census
gofEvTC <- sienaGOF(
  modelEv,
  verbose = FALSE,
  varName = "friendship", TriadCensus,
  cluster = cl
)
```

```r
# Geodesic distance
GeodesicDistribution <- function(
    i, data, sims, period, groupName,
    varName, levls = c(1:5, Inf), cumulative = TRUE
) {
  x <- networkExtraction(i, data, sims, period, groupName, varName)
  require(sna)
  a <- sna::geodist(symmetrize(x))$gdist
  if (cumulative) {
    gdi <- sapply(levls, function(i) {
      sum(a <= i)
    })
  }
  else {
    gdi <- sapply(levls, function(i) {
      sum(a == i)
    })
  }
  names(gdi) <- as.character(levls)
  gdi
}

gofEvGD <- sienaGOF(
  modelEv,
  verbose = FALSE,
  varName = "friendship", GeodesicDistribution
)


plot(gofEvId)
```

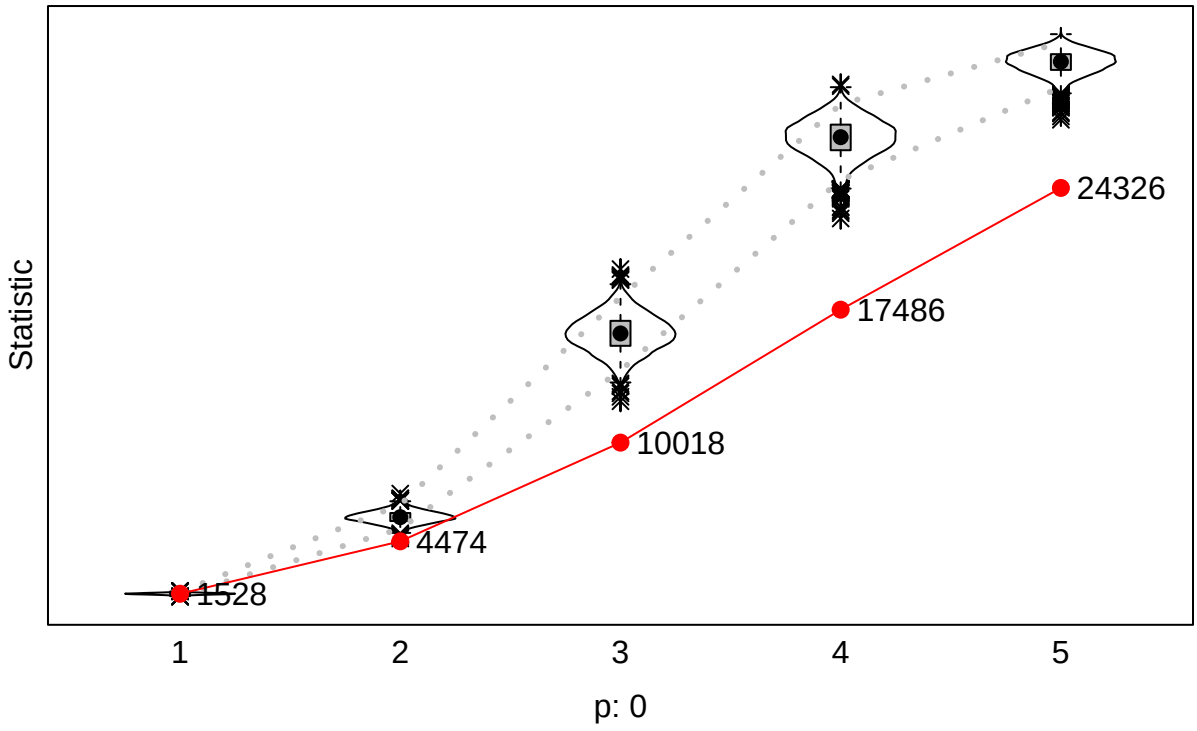



```
plot(gofEvOd)
```

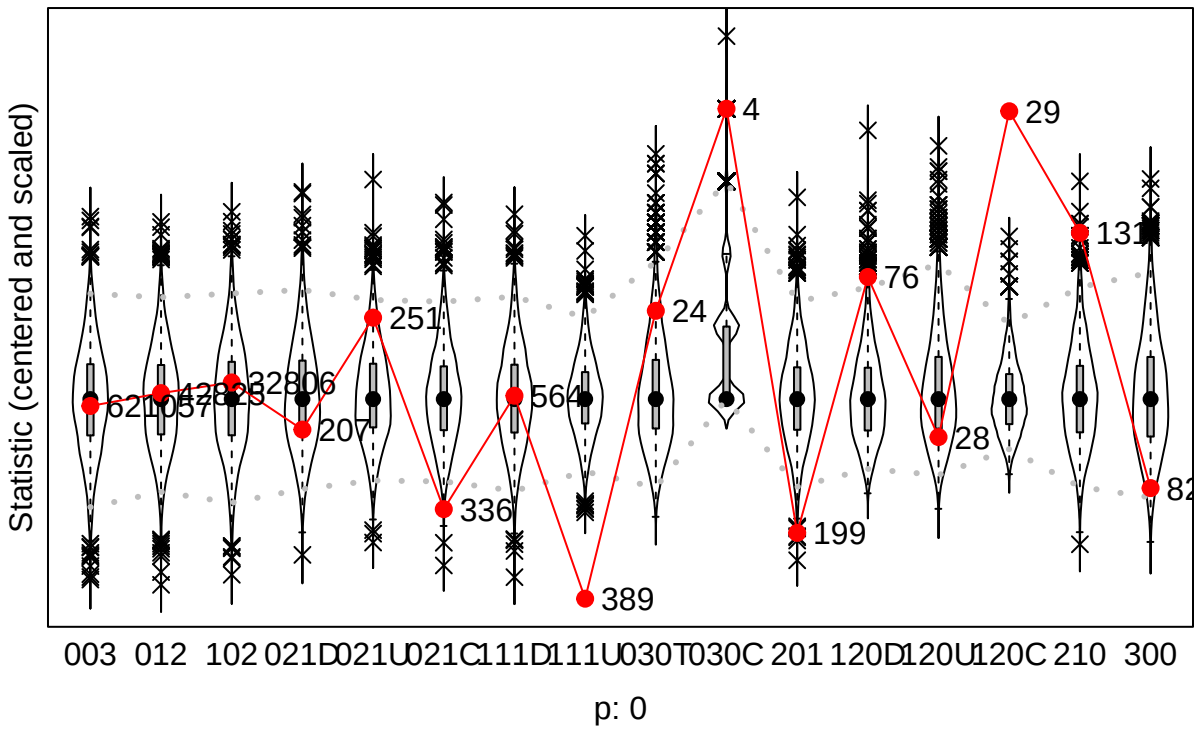



```
plot(gofEvGD)
```

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




```
plot(gofEvTC, center = TRUE, scale = TRUE)
```



Even though the model has converged, the fit is bad, because, as we can see from all the plots except the one related to the indegree distribution, the observed values are not all inside the 95% confidence intervals.

```
printSiena(modelEv)
```

```
##     dependent                                effect  theta  s.e. p.value sig.
## 1        rate constant friendship rate (period 1) 10.868 1.011
## 2        rate constant friendship rate (period 2)  8.912 0.800
## 3  friendship                  outdegree (density) -2.396 0.092       0  ***
## 4  friendship                         reciprocity  2.432 0.107       0  ***
## 5  friendship                transitive triplets  0.751 0.047       0  ***
## 6  friendship                           3-cycles -0.589 0.092       0  ***
## 7  friendship               indegree - popularity -0.088 0.020       0  ***
## 8  friendship                         logdistance -0.172 0.047       0  ***
## 9  friendship                       alcohol alter  0.032 0.031   0.297
## 10 friendship                         alcohol ego -0.013 0.029    0.66
## 11 friendship                 alcohol similarity  0.639 0.159       0  ***
##    t.conv
## 1
## 2
## 3   0.002
## 4  -0.016
## 5       0
## 6  -0.011
## 7   0.004
## 8  -0.008
```

```
## 9   0.011
## 10 -0.017
## 11  0.057
```

All the included parameters are significant, except for the control effects (alcohol ego and alcohol alter). The density parameter is negative, as we expected from a social network (sparse network). The positive reciprocity parameter suggests that an actor tends to reciprocate a friendship nomination. The negative logdistance parameter suggests that there is a negative correlation between distance between students and tendency to form a tie. The negative indegree popularity suggests that there is a tendency to nominate as a friend unpopular students (low in-degree). The positive alcohol similarity effect suggests that students with similar alcohol consumption tend to be friends.

(1.4) The first hypothesis is not supported by data, because the parameter related to indegree popularity is negative. The second hypothesis is also supported by data, so we can say that students tend to be friends with pupils with similar alcohol consumption to their own. Since the estimate of the coefficient related to the logdistance is significant and negative, we can say the smaller the distance, the more likely the students are to be friends, so we do not have evidence to reject hypothesis 3.

(2) (2.1)

```
alcoholbeh <- sienaDependent(alcohol, type = "behavior")
mydata <- sienaDataCreate(friendship, alcoholbeh, gender, age, logdistance)
print01Report(mydata, modelname = "Glasgowbeh")
# Moran index
moran1 <- nacf(f1, alcohol[, 1], lag.max = 1, type = "moran",
               neighborhood.type = "out", mode = "digraph")
moran2 <- nacf(f2, alcohol[, 2], lag.max = 1, type = "moran",
               neighborhood.type = "out", mode = "digraph")
moran3 <- nacf(f3, alcohol[, 3], lag.max = 1, type = "moran",
               neighborhood.type = "out", mode = "digraph")
autocorr <- rbind(moran1, moran2, moran3)
autocorr[,2]
```

```
##    moran1    moran2    moran3
## 0.2450651 0.3394460 0.3316509
```

We can see that there is positive autocorrelation between the friendship ties and the alcohol consumption.

```
myeff <- getEffects(mydata)
myeff <- includeEffects(myeff, transTrip)
```

```
##   effectName          include fix   test  initialValue parm
## 1 transitive triplets TRUE    FALSE FALSE          0   0
```

```
myeff <- includeEffects(myeff, inPop)
```

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

```
myeff <- includeEffects(myeff, egoX, altX, simX, interaction1='alcoholbeh')
```

```
##   effectName            include fix   test  initialValue parm
## 1 alcoholbeh alter      TRUE    FALSE FALSE          0   0
## 2 alcoholbeh ego        TRUE    FALSE FALSE          0   0
## 3 alcoholbeh similarity TRUE    FALSE FALSE          0   0
```

```
myeff <- includeEffects(myeff, X, interaction1='logdistance')
```

```
##   effectName  include fix   test  initialValue parm
## 1 logdistance TRUE    FALSE FALSE          0   0
```

```
myeff <- includeEffects(myeff, indeg, avSim,
                        name = "alcoholbeh", interaction1 = "friendship")
```

```
##   effectName                   include fix   test  initialValue parm
## 1 alcoholbeh average similarity TRUE    FALSE FALSE          0   0
## 2 alcoholbeh indegree           TRUE    FALSE FALSE          0   0
```

(2.2)

```
myAlgorithm <- sienaAlgorithmCreate(
  projname = "CoevGlasgow",
  nsub = 4, n3 = 3000, seed = 2023
)
```

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

```
# Model estimation
modelCoev <- siena07(
  myAlgorithm,
  data = mydata, effects = myeff,
  returnDeps = TRUE, batch = FALSE,
  useCluster = TRUE, nbrNodes = 4
)
modelCoev
```

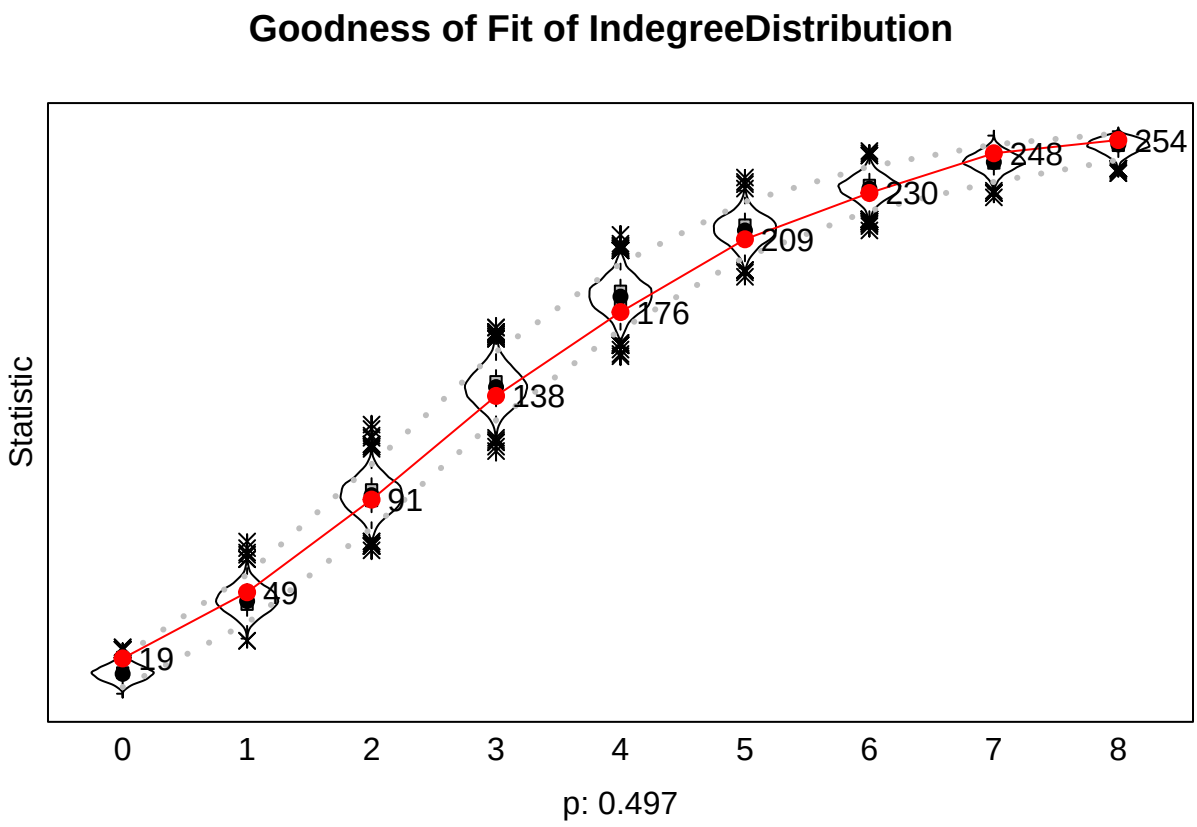
```
## Estimates, standard errors and convergence t-ratios
##
##                                                 Estimate   Standard   Convergence
##                                                              Error      t-ratio
## Network Dynamics
##    1. rate constant friendship rate (period 1) 10.7278  ( 1.0139   )   -0.0435
##    2. rate constant friendship rate (period 2)  8.8263  ( 0.7622   )    0.0409
##    3. eval outdegree (density)                 -2.4138  ( 0.0970   )   -0.0032
##    4. eval reciprocity                          2.3948  ( 0.1021   )    0.0063
##    5. eval transitive triplets                  0.7422  ( 0.0455   )    0.0009
##    6. eval 3-cycles                            -0.5730  ( 0.0892   )    0.0019
##    7. eval indegree - popularity               -0.0880  ( 0.0206   )    0.0032
##    8. eval logdistance                         -0.1780  ( 0.0458   )   -0.0918
```

```
##    9. eval alcoholbeh alter                     0.0414  ( 0.0473   )    0.0015
##   10. eval alcoholbeh ego                      -0.0277  ( 0.0444   )   -0.0014
##   11. eval alcoholbeh similarity                1.2097  ( 0.3678   )    0.0283
##
## Behavior Dynamics
##   12. rate rate alcoholbeh (period 1)           1.6303  ( 0.2640   )   -0.0260
##   13. rate rate alcoholbeh (period 2)           2.3414  ( 0.4350   )    0.0098
##   14. eval alcoholbeh linear shape              0.1412  ( 0.2471   )    0.0104
##   15. eval alcoholbeh quadratic shape           0.0418  ( 0.0669   )   -0.0109
##   16. eval alcoholbeh average similarity        7.1753  ( 2.0503   )    0.0139
##   17. eval alcoholbeh indegree                  0.0717  ( 0.0677   )    0.0261
##
## Overall maximum convergence ratio:    0.1365
##
##
## Total of 3983 iteration steps.
```

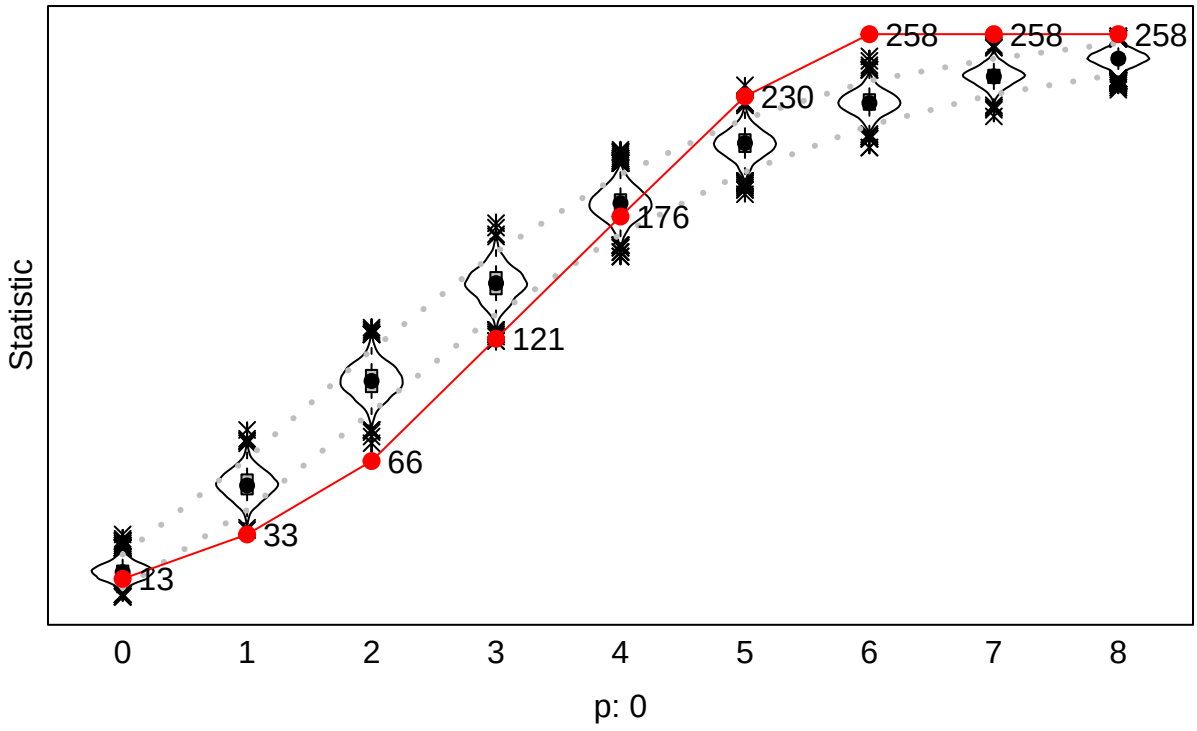
The algorithm converged, because the absolute value of the maximum t-ratio of all the parameters is below 0.1 and the overall maximum convergence ratio is below 0.2.

```r
cl <- makeCluster(4)

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

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

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


# Geodesic distance
gofCoevGD <- sienaGOF(
  modelCoev,
  verbose = FALSE,
  varName = "friendship", GeodesicDistribution
)
```

```
# Behaviour distribution
gofCoevBeh <- sienaGOF(
  modelCoev,
  verbose = FALSE,
  varName = "alcoholbeh", BehaviorDistribution,
  cluster = cl
)

stopCluster(cl)

plot(gofCoevId)
```

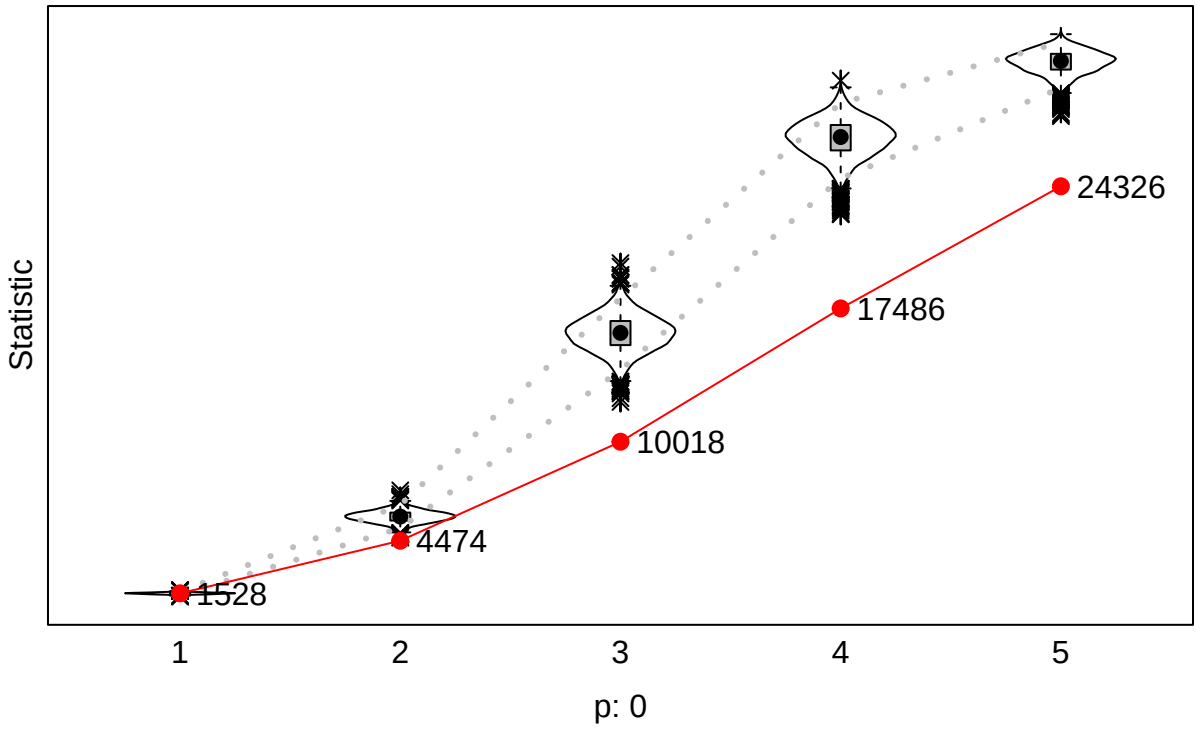



```
plot(gofCoevOd)
```

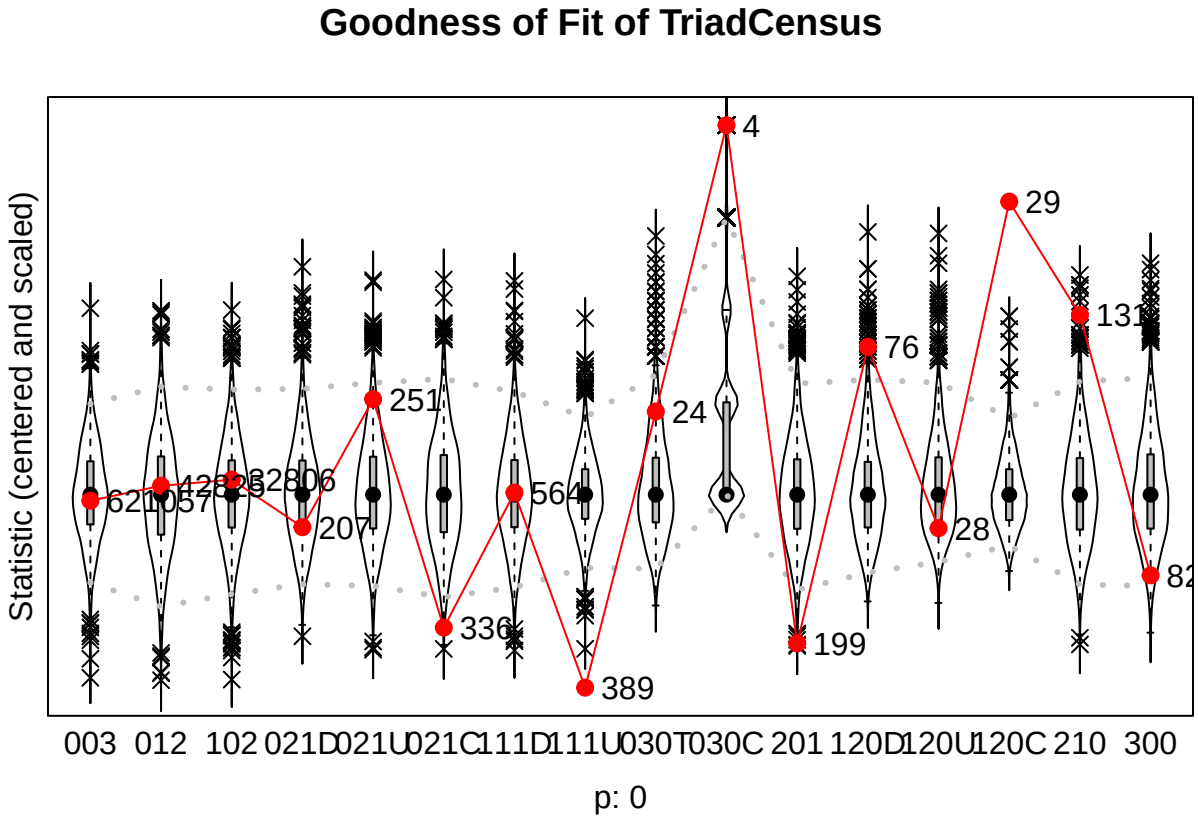



```
plot(gofCoevGD)
```

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

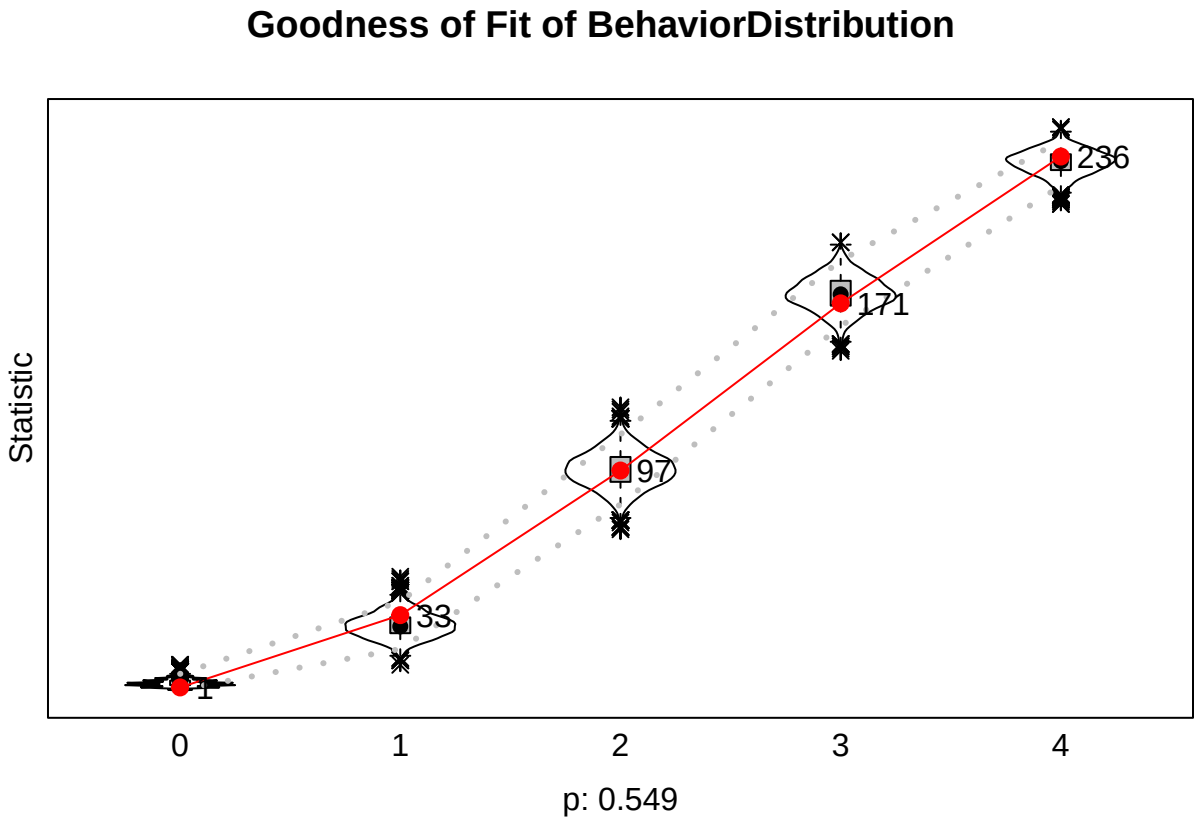



```
plot(gofCoevTC, center = TRUE, scale = TRUE)
```




```
plot(gofCoevBeh)
```

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



Even when investigating the co-evolution of friendship and alcohol consumption, the model has a bad fit, because the outdegree distribution, the geodesic distribution and the triad census are not well represented by the model.

```
printSienaCoev(modelCoev)
```

```
##                                effect  theta  s.e. p.value sig. t.conv
## 1  constant friendship rate (period 1) 10.728 1.014
## 2  constant friendship rate (period 2)  8.826 0.762
## 3                  outdegree (density) -2.414 0.097       0  *** -0.003
## 4                         reciprocity  2.395 0.102       0  ***  0.006
## 5                 transitive triplets  0.742 0.046       0  ***  0.001
## 6                           3-cycles -0.573 0.089       0  ***  0.002
## 7               indegree - popularity -0.088 0.021       0  ***  0.003
## 8                         logdistance -0.178 0.046       0  *** -0.092
## 9                    alcoholbeh alter  0.041 0.047   0.382       0.002
## 10                    alcoholbeh ego -0.028 0.044   0.533      -0.001
## 11              alcoholbeh similarity  1.210 0.368   0.001   **  0.028
## 12         rate alcoholbeh (period 1)  1.630 0.264
## 13         rate alcoholbeh (period 2)  2.341 0.435
## 14            alcoholbeh linear shape  0.141 0.247   0.568        0.01
## 15         alcoholbeh quadratic shape  0.042 0.067   0.532      -0.011
## 16      alcoholbeh average similarity  7.175 2.050       0  ***  0.014
## 17                alcoholbeh indegree  0.072 0.068   0.289       0.026
```

The estimates of the parameters shared with the previous models lead to the same conclusions. The non-

significant parameter for linear shape suggests that there is no tendency to either decrease or increase alcohol consumption in general. The non-significant parameter for quadratic shape suggests that there is no tendency to form a U or inverse U shape over time. The significant positive parameter for alcoholbeh similarity suggests that students are more likely to form friendship ties to students with similar drinking habits. The significant positive parameter for alcoholbeh average similarity suggests that students adjust their alcohol consumption to that of their friends. There is no evidence that popular students tend to increase or maintain their level of alcohol consumption.

(2.3) The fourth hypothesis is not supported by the data because the coefficient related to alcoholbeh indegree is not significant. The fifth hypothesis is supported by the data, so we can say that students tend to adjust their alcohol consumption to that of their friends. The conclusions about the hypotheses in (1) do not change.

(2.4) We have evidence for both selection and influence processes. As an example, hypothesis 2 is related to the selection process and we found evidence to support it. Similarly, hypothesis 5 is related to an influence process (students adjusting their drinking behavior) and we found evidence to support it.

(3) Since the outdegree distribution is not well represented by the model, comparing the outdegree distribution in the observed network to the one in the simulated networks, the first results to be more skewed, so it may be necessary to let the rate function depend on the outdegrees, as specified in the RSiena manual. Additionally, another way to improve the representation of the outdegree statistic would be to use different outdegree variables for different types of individuals (e.g. individuals with different alcohol consumption habits) instead of just one variable for all individuals.

As regards the geodesic distances, in the simulated networks they are too large compared to the observed network. Hence, we might consider adding the gwesp effect, which is an alternative expression for transitivity. By doing so, we can potentially increase transitivity and thus shorten the distance between nodes.