Extensions to Social Network Analysis for ERGM in Multiplex Networks: A Tutorial Paper

Tyler Maule

December 7th, 2021

Abstract

Over the course of this tutorial paper, readers expand their understanding of exponential random graph models in social network analysis to encompass the conceptual understanding and usage of multiplex networks. Motivated by the complexity of social interrelationships, multiplex networks offer a means of modeling multiple distinct pairwise relationships as edges in separate layers. After a review of standard notation, readers encounter these networks within the framework of multilayer networks. They expand on network definitions, substructures, measures, and representations. With this groundwork complete, the reader appreciates the process of redefining, fitting, and interpreting ERGMs in the multiplex setting. The paper then surveys pertinent software packages in R, culminating in a example application.

Introduction & Motivation

At any level, a full understanding of social dynamics requires knowledge of not only individual actors, but of the complex interrelationships between them. Network science represents a critical approach to developing this knowledge. Most generally, networks are conceived as a collection of nodes and a series of edges between those nodes. Ordinary graphs in mathematics exclude directed edges, different types of edges, self-loops, and multiple edges (Nordhaus, Stewart 1963).

In contrast, social landscapes induce, and are induced by, a great diversity of interactions and interconnections. Residents of any large city may be represented as nodes in a network. Edges linking pairs of those nodes could indicate that the residents are friends, immediate family, coworkers, neighbors, classmates, or mutually connected by any number of relationship types. Alternatively they could indicate that the pair share something: the same voting record, the same workday commute, the same preferences in partners. Of course, the edges could signal incidental connections, such as making eye contact in a crowded mall. We may then be curious about any connections between those edge types. Are individuals who frequent the same set of restaurants more likely to consume media from the same set of sources than those who frequent different restaurants?

Needless to say, statistical inference informing complex research questions necessitates the representation of multiple types of edges in the same network structure. These questions can be addressed with the use of multiplex networks, defined here as networks in which the same set of nodes can be connected with distinct edge types on distinct *layers*. While we use examples from the social sciences here, multiplex networks have a wide range of applications across disciplines.

Defining Multiplex Networks

Given the multitude of compelling reasons to use multiplex networks, we turn to the challenge of representing them mathematically. Goals from visualization to statistical inference require that multiplex networks be represented by formal data structures. The optimal choice of data structure will enable scalable computation while remaining general enough to address a great number of contexts. Traditionally, representations of multiplex networks expand on the formulation of monoplex networks.

Define V as a set of nodes and $E \subseteq V \times V$ as a set of edges. Consider a group of n schoolchildren, some of whom are friends with one another. We could represent their friendship relations as a set of edges, with up to *n(n-1)/2* edges if we assume friendships are reciprocal and *n(n-1)* otherwise. Then the tuple $G = (V, E)$ defines a monoplex friendship network amongst the children. E can form the basis of an adjacency matrix, in

which the presence or absence of each potential pairwise edge is captured by a binary n by n matrix.

Kivela et. al. advance a framework of *multilayer* networks to organize the proliferation of novel network structures studied in the 20th and early 21st centuries (Kivela et. al., 2014). Their framework retains the structure of a set V that contains all nodes in the network. In this case, however, each node belongs to one or more layers. Different layers may represent different types of interactions between actors: say, friends and coworkers. Layers can organize different types of products in co-purchase networks; for example, one layer could hold edges between actors that purchased the same car, the second layer could hold edges between actors that purchased the same car insurance, and the third could capture co-purchases based on the dealership each car was purchased at. If we wanted to represent all of the examples mentioned as distinct layers in the same network, we could incorporate one *aspect* of layers pertaining to car-related purchases and a second aspect of interaction-based layers. As one might guess, aspects refer to non-overlapping subsets of a network's overall set of layers . If we consider a single layer in one aspect without respect to any other aspects, we consider it an *elementary layer*. In actuality, we usually consider nodes to belong to a layer produced by interaction of all aspects. So if we want to look at Isabella's friends who bought the same car as her, we'd study the layer at the "friends" layer of the interaction aspect and the "car purchased" level of the co-purchasing aspect. It follows that we can consider the group of all layers as a set defined by the cartesian product of all aspects.

For a mathematical representation of this toy example, let the co-purchasing aspect be $L_1 = \{C, I, D\}$ where C refers to the car co-purchasing layer, I the insurance related layer and D the dealership-related layer. Let the interactions aspect be $L_2 = \{F, W\}$ where F connects actors through friendship and W connects those who work together. Then the set of all layers in our example network is $L = L_1 \times L_2 = \{(C, F), (C, W), (I, F), (I, W), (D, F), (D, W)\}$. Generally, if a multilayer network has $d \in \mathbb{Z}^+$ aspects where aspect a has set of layers L_a, the set of all elementary layers is $\mathbf{L} = \{L_1, L_2, \ldots, L_d\}$ and the set of all layers is $L = L_1 \times L_2 \times \cdots \times L_d$.

At this open-ended stage, we express that a given layer (C, F) contains a given node w using a node-layer tuple (u, C, F). The set of all node-layer tuples that exist in a given network is given as $V_M \subseteq V \times L_1 \times L_2 \times L_d$. If we define hyperedges as edges that connect $k \in \{2, 3, \ldots, |V|\}$ nodes, the set of edges could be denoted $E_M \subseteq V_M^k = \Pi_{i=2}^k V_M$. Multiplex networks as defined by Kivela et. al., however, only include pairwise edges and hence we denote $E_M \subseteq V_M \times V_M$. Note that *intralayer edges* connect nodes on the same layer, whereas *interlayer edges* tie nodes across layers.

Altogether a multilayer network takes the general form $M = (V, V_M, E_M, \mathbf{L})$. From the adjacency matrix of the monoplex network we can easily conceive of the multiplex network's *adjacency tensor* based on

E_M. Define the adjacency tensor by letting $\mathcal{A} = \{0,1\}^{|V|^2 \times |L_1|^2 \times \cdots \times |L_d|^2}$. If we wish to determine whether node u in layer $\alpha \in \mathbf{L}$ has a tie with node v in layer $\beta \in \mathbf{L}$ we inspect $\mathcal{A}$ at the index of u's node-layer tuple concatenated by v's node-layer tuple. That is, we check the value at index (u, α, v, β) of $\mathcal{A}$. Following traditional notation, a value of *1* conveys that the relevant edge does indeed exist.

To facilitate computation for network analysis, the adjacency tensor must be transformed into a data structure that is easier to work with. Potential transformations include aggregations, wherein layers are treated as labels of nodes in a monoplex network. Flattening refers to the merging of two or more aspects, simplifying the network structure through a bijective mapping. These methods, however, result in information loss and limit the depth of analysis possible.

By mapping the adjacency tensor into a block matrix with partitions, we can apply many of the analytic methods used for monoplex networks (Chen 2019). Say that we represent a single layer α_i, $i = 1, \ldots, m$ where $m = |L_1| \times |L_2| \times \cdots \times |L_d|$, as a standard monoplex adjacency matrix A_{ii} communicating all edges within layer α_i. Now take A_{ij} to be a standard adjacency matrix with all edges between distinct layers α_i and α_j, $j = 1, \ldots, m$. For a directed matrix, A_{ij} would represent edges from nodes in layer α_i to those in α_b, and the opposite direction would follow similarly as A_{ji}. Then we can construct a m by m matrix $\mathbf{A}$ where entry $\mathbf{A}_{ij}$ of this new matrix is the entire standard adjacency matrix A_{ij}. Thus the diagonal of $\mathbf{A}$ contains information on intralayer edges, and the off-diagonal entries correspond to interlayer edges. This representation, known as a *supra-adjacency matrix*, necessarily loses information on the relations between different layers and their underlying aspects. From this point forward, we categorically minimize our focus on aspects.

After we construct a suitable data structure to contain the network, we can consider whether to place any constraints on the structure. In some contexts the appropriate constraints will simplify the supra-adjacency matrix. For example, if we prohibit intralayer edges the diagonal of $\mathbf{A}$ will contain only zero matrices. Undirected multiplex networks will naturally have symmetric supra-adjacency matrices. If we consider nodes on different layers to have different types and disallow intralayer edges, we have a *node-colored* network.

Based on this broad framework of multilayer networks, we can build our understanding of multiplex networks. Critically important is the restriction that all nodes must appear in all layers, rather than a subset. A pair of nodes may be connected by an edge on one layer and disconnected on another, but their omnipresence is key. Since each layer contains the same set of actors, we can imagine that an implicit interlayer edge connects a given node from layer to layer. Nevertheless, these implicit edges are not represented by the network data structure. In fact, multiplex networks contain no interlayer edges.

Due to these restrictions, we can simplify the multilayer definitions to reflect multiplex networks

specifically. Recall that we defined multilayer networks as $M = (V, V_M, E_M, \mathbf{L})$. Here $V_M = V$, so we can drop the term V_M from the quadruple. Since we only permit intralayer ties, the supra-adjacency matrix has zero matrices in off-diagonal spaces. E_M, it follows, takes a simpler form as well. We are still left with the triplet $M = (V, E_M, \mathbf{L})$, but our supra-adjacency matrix and resulting analyses are narrowed in a sense. Hence the multiplex network takes form as a special class of multilayer networks.

Metrics & Motifs in Multiplex Networks

At this point we have aligned our network notation, detailed the framework of multilayer networks, and moved to define multiplex networks. Having established this definition and a suitable data structure, we turn to focus on multiplex network metrics and motifs. Like in the monoplex case, these descriptors can be used to characterize, compare, and model networks. Many of the relevant descriptors extend from the monoplex case directly, while others incorporate cross-layer relations.

Note that the set of metrics described below is far from exhaustive. Instead, this section aims to illustrate the process of generalizing monoplex measures and introduce some of the many potential statistics used in modeling processes.

We begin by generalizing from local metrics in the monoplex case. Recall that *degree* refers to the number of edges that a given node shares with other nodes, whether incoming, outgoing, mutual, or undirected. That node's *neighborhood* denotes the set of all other nodes with which it shares an edge. Now by summing the number of edges that a node has in each layer, we can find that node's *aggregate degree*, and its *aggregate neighborhood* follows similarly (Kivela et. al., 2014). When examining groups of layers, one can aggregate degree or neighborhood statistics across a subset of the network's layers (Berlingerio et. al., 2011). Alternatively, we could set a threshold: a node must share at least x edges with another node across all layers to be counted towards the x-thresholded degree or x-thresholded neighborhood (Kivela et. al., 2014).

In some cases we might wish to standardize the notion of degree, perhaps specifying that a pair of nodes must have an edge in a given proportion of layers to count, using quantiles of edge counts, or differential weighting of edges on different layers. Furthermore, ratios of nodes' degrees over a subset of layers to their aggregate degrees can be used to gauge the redundancy of that layer subset to the nodes' degrees (Berlingerio et. al., 2013). Comparisons of the number of edges on at least one layer in a subset of layers to the number of edges present in all layers of that subset can also yield a measure of layer redundancy that (Berlingerio et. al., 2013) term the Pair D-Correlation.

Analysis of path lengths and intralayer walks proceed similarly. First consider that the path length

between two given nodes u and v as well as the geodesic may differ from layer to layer. Calculating these metrics for each layer will yield a vector with l values for a multiplex network with l layers. Then if they wish, the analyst can produce a single metric with a function that somehow aggregates values in said vector (Kivela et. al., 2014).

One can imagine that many other monoplex metrics can easily be generalized to the multiplex case by evaluating the metric layer-by-layer and applying an aggregation function if desired. This process extends beyond local metrics to encompass the comparison of global monoplex metrics between layers. For example, (De Domenico et. al. 2014) measure the distribution of node eigenvector centralities in each layer and compare these distributions across layers using annular visualizations.

Given our ultimate goal of modeling multilplex networks, we now exemplify the proliferation of network motifs. Like the metrics introduced earlier, motifs can be specified on a layer-by-layer basis. With a network of l layers, we may count alternating k-stars for each layer as we would in the monoplex setting, retrieving l distinct terms as a result. Substitute k-stars for any other monoplex motif and it is simple to grasp the increased array of useful layerwise motifs.

Cross-layer motifs continue to distinguish the multiplex case. For instance, the idea of pairwise reciprocal connections between nodes can be altered to incorporate cross-layer forms of reciprocity (Chen 2019). Using a subset of two layers, we note that reciprocity on one layer may occur on the second, but could instead be accompanied by an unreciprocated directed tie on that second layer. Along with unreciprocated ties from node i to j on both layers or a tie from i to j on one layer paired with a tie from j to i on the second, the aforementioned pairwise reciprocal ties form a *duplex dyad census* of potential connection configurations involving a pair of layers. This collection of five pairwise motifs is archetypical of new motifs that capture cross-layer relations. Naturally, the census can be extended to subsets of three or more layers. Comparable censuses can be undertaken on most other monoplex motifs to develop additional potential configurations.

Evidently, moving from G to G_M brings a host of additional summary statistics and network substructures that facilitate the understanding of multiplex networks.

ERGM in Multiplex Networks

With the groundwork of multiplex network definitions and metrics established, we turn to the core topic of this tutorial paper: the use of models to conduct statistical inference in the multiplex setting. While there exist a wide variety of pertinent dynamic, temporal, generative, community detection, and attribute-focused models, here we focus primarily on exponential random graph models (ERGMs).

Initially known as Markov graphs, ERGMs were developed to relate the "sociometric properties of networks" in their use of log-linear family models (Frank, Strauss 1986). ERGMs view an empirically given network as a realization from a probability distribution of potential networks given some constraints. They consider network motifs, or configurations, as sufficient statistics of this network distribution (Ghafouri Khasteh 2020). Then with an extension of logistic regression to accommodate the dependence structure of a network and its motifs, the network distribution may be modeled by using motifs as explanatory covariates. Here we instantiate an ERGM model as:

$$P_G(X = x|\theta) = (\frac{1}{Z(\theta)}) \cdot exp\{\theta^T \mathbf{f}(G)\}$$

where x represents the realized network, X is the set of all potential networks under predefined constraints, $P(X = x|\theta)$ denotes the probability of realizing that network under the distribution of networks P_G given a parameter vector θ, and $\mathbf{f}(G)$ is a vector of functions defined on a network G that return network motif counts. Finally, $Z(\theta)$ is a normalizing constant:

$$Z(\theta) = \sum_{x \in X} exp\{\theta^T \mathbf{f}(G)\}$$

At its simplest, moving from the monoplex case to the multiplex case requires extensions to the selection of network motifs included through $\mathbf{f}(G)$ and modification of our probability distribution P_G (Chen 2019). Earlier we outlined a selection of new metrics based suitable for multiplex networks, say, $\mathbf{f}'(G_M)$. Next we will elaborate methods for estimation based on the network probability distribution, to be denoted P_{G_M}. Call the reexpression of the normalizing constant $Z'(\theta)$, as it now relies on $\mathbf{f}'(G_M)$ rather than $\mathbf{f}(G)$. Then the expression of this novel ERGM takes the subtly different form:

$$P_{G_M}(X = x|\theta) = (\frac{1}{Z'(\theta)}) \cdot exp\{\theta^T \mathbf{f}'(G_M)\}$$

Like parameter estimation in the monoplex setting, multiplex ERGMs rely on maximum likelihood estimation for model fitting. They seek to find $\hat{\theta}_{MLE}$ which maximizes P_{G_M} over the parameter space Θ. For exponential family distributions in the simplest statistical cases, it is appropriate to set the partial derivative of the log-likelihood function given each parameter of interest to zero and maximize accordingly. When closed-form solutions are intractable, we turn to methods of statistical computation including Markov Chain Monte Carlo (MCMC) sampling. Typically ERGM fitting makes use of Gibbs sampling or the Metropolis-Hastings (M-H) MCMC algorithm (Chatterjee, Diaconis 2013). Alternatives include Robbins-Monro optimization and

psuedo-likelihood methods for graphs, as introduced by (Besag 1975).

In dealing with networks, M-H MCMC initializes a random network based on provided constraints. Here, make use of the constraints defined on the supra-adjacency matrix. That is, each layer will contain the same number of nodes and interlayer edges are prohibited. For the directed case, the random supra-adjacency matrix will be an asymmetric binary matrix with zero matrices on the off-diagonal adjacency matrices.

Constraints established, the algorithm iteratively generates proposal networks x^* by randomly adding or removing an edge in the network at step t, denoted $x^{(t)}$ (Ghafouri, Khasteh 2020). The proposal network's acceptance ratio R hinges on the comparison of $P_{G_M}(X = x^*|\theta)$ with $P_{G_M}(X = x^{(t)}|\theta)$. Importance sampling and stochastic sampling, both implemented according to the Newton-Raphson method, often complement the MCMC algorithm in maximizing P_{G_M} over the parameter space. Iteration halts upon convergence, usually measured at the $\alpha = 0.01$ level of significance. A much more in-depth discussion of estimation procedures for monoplex networks can be found in (Lusher, Koskinen, Robins 2012), which generalizes to multiplex networks based on the use of the defined supra-adjacency matrix.

Goodness-of-fit measures follow naturally from the standard ERGM procedures. One should inspect the MCMC trace plots to ensure convergence and simulate additional matrices from the fitted model for evaluating goodness-of-fit. Tools to gauge goodness-of-fit have not yet been implemented in R, as the existing implementations fail to distinguish between different layers in their calculation of statistics based upon simulated networks.

We have redefined the mathematical expression of ERGM, described the model estimation process, and briefly introduced means for evaluating the fitted model. Together with the outline of new network motifs and metrics, we have offered a general survey of tools for analyzing multiplex networks. Now we turn to the implementation of these tools in the statistical programming language R.

R Packages for Multiplex Network Analysis

Network visualizaton remains a key component of understanding and communicating multiplex network structures. It is straightforward to understand cross-layer relations with visualization packages like `multinet` (Magnani et. al., 2021) and `muxViz` (De Domenico, 2015), which focus on multilayer networks. Both packages, however, lack a polished design and flexible layout options. Since the only interlayer ties in a multiplex network are implicit, one can easily visualize each layer separately with popular software packages such as `igraph` (Csárdi 2020), `ggnetwork` (Tyner et. al. 2017), and `plotly` (Plotly Technologies Inc., 2015). `igraph` allows users to provide a fixed position for each node. Therefore, at this time we recommend use of `igraph`

despite its limitations.

For the computation of metrics on multiplex networks, `multinet` and `muxViz` do currently provide the most comprehensive suite of methods for use with R. Still, these packages focus on techniques for centrality calculation, community detection, distance measurement, and generative models rather than the motifs or models underlying ERGM. In addition, neither package is fully implemented and `muxViz` lacks clear documentation.

We recommend `multilayer.ergm` for working with network motifs and estimating ERGM models (Chen 2019). At the time of this writing, there are no alternative R packages designed with multiplex ERGMs in mind. With eleven intralayer terms and four interlayer terms implemented, the package offers a limited but useful set of potential ERGM parameters. Current interlayer terms deal primarily with two-layer networks. These terms, as well as the model estimation function, build on the `ergm` package from `statnet` (Morris et. al. 2008). Consequently, the MCMC fitting algorithm is not optimized for use with multiplex networks, and the goodness-of-fit metrics do not take multiple layers into account. Regardless, `multilayer.ergm` serves well for modeling smaller networks with few layers and a core set of model parameters.

Finally, the `mlergm` package developed for multilevel exponential-family random graph modeling provides an alternative for modeling two-layer multiplex networks. While multilevel techniques usually fit data with a hierarchical structure, we may treat one layer as the higher level and the other as the lower layer. Each layer should contain all nodes. Then the network that communicates each lower-level actor's affiliation with higher-level entities, often termed the meso-level network, can be used to explicitly model the implicit ties from an actor in one layer to the same actor in the second layer (Wang et. al., 2013).

An Application of Multiplex Networks in R

To round out this paper, we present an application of the `multilayer.ergm` package to multiplex data. The data used is sourced from the study "Changing climates of conflict: A social network experiment in 56 schools" by (Paluck et. al. 2016). Researchers conducted a yearlong study in fifty-six New Jersey middle schools in which randomly selected students from randomly assigned treatment schools received regular anti-conflict trainings. At the start and conclusion of the study, all students were administered a survey that gauged demographic information, behavioral records, values related to social engagement, and interactions amongst students. Students were asked to nominate up to ten classmates with whom they spent time, five classmates with whom they experienced conflict, and two classmates that they considered best friends.

We used the results from the first survey to construct a multiplex network of students in one school.

The school with ID 15 gathered survey results from 103 students, who serve as nodes in this application's network. Directed edges represent nominations of one student by another, such that edges run from the nominator to the nominee. One layer contains 738 ties related to time spent together, the second contains 259 ties related to conflict relationships, and the third contains 193 ties corresponding to friendship relations. Each node has attribute data on students' genders, assigned treatment group, ethnicity, and personal beliefs.

Prior to analysis in R, we developed three hypotheses:

(1) Boys will be more likely than girls to be nominated as involved in a conflict relationship, all else being equal.

(2) If a student nominates a peer as spending time with them, the nominated person will be more likely to nominate the first student than if they were not nominated, all else being equal.

(3) If two students nominate each other as involved in a friendship relationship, it will be less likely that one of those students will nominate the other as involved in a conflict relationship than if the two students did not nominate each other as involved in a friendship relationship, all else being equal.

For readers interested in the full example analysis, we include an R Markdown export in Appendix I. Below we annotate a portion of that analysis.

Using `dplyr` and `igraph`, we filtered data from Paluck et. al. and produced a list of student IDs and lists of those students' nominations corresponding to each category. Each layer was created as a monoplex network. In output figures (1)-(3) below see visualizations of the three layers in our multiplex model, each produced with `igraph`. Students assigned to treatment are highlighted in pink and students assigned to control appear in blue.

Figure (1) Layer I. Arrows from one student to another indicate that the first student named the second when prompted, "In the last few weeks I decided to spend time with these students at my school: (in school, out of school, or online)" (Paluck et. al. 2016)

Figure (2) Layer II. Arrows from one student to another indicate that the first student named the second when prompted, "I have conflict with these students who go to my school: (face to face, texts, online)" (Paluck et. al. 2016)

Figure (3) Layer III. Arrows from one student to another indicate that the first student named the second when prompted, "This is my best friend (or, these are my two best friends) at this school:" (Paluck et. al. 2016)

With the aim of simplifying this example, we restrict our attention to layers I and II in the modeling

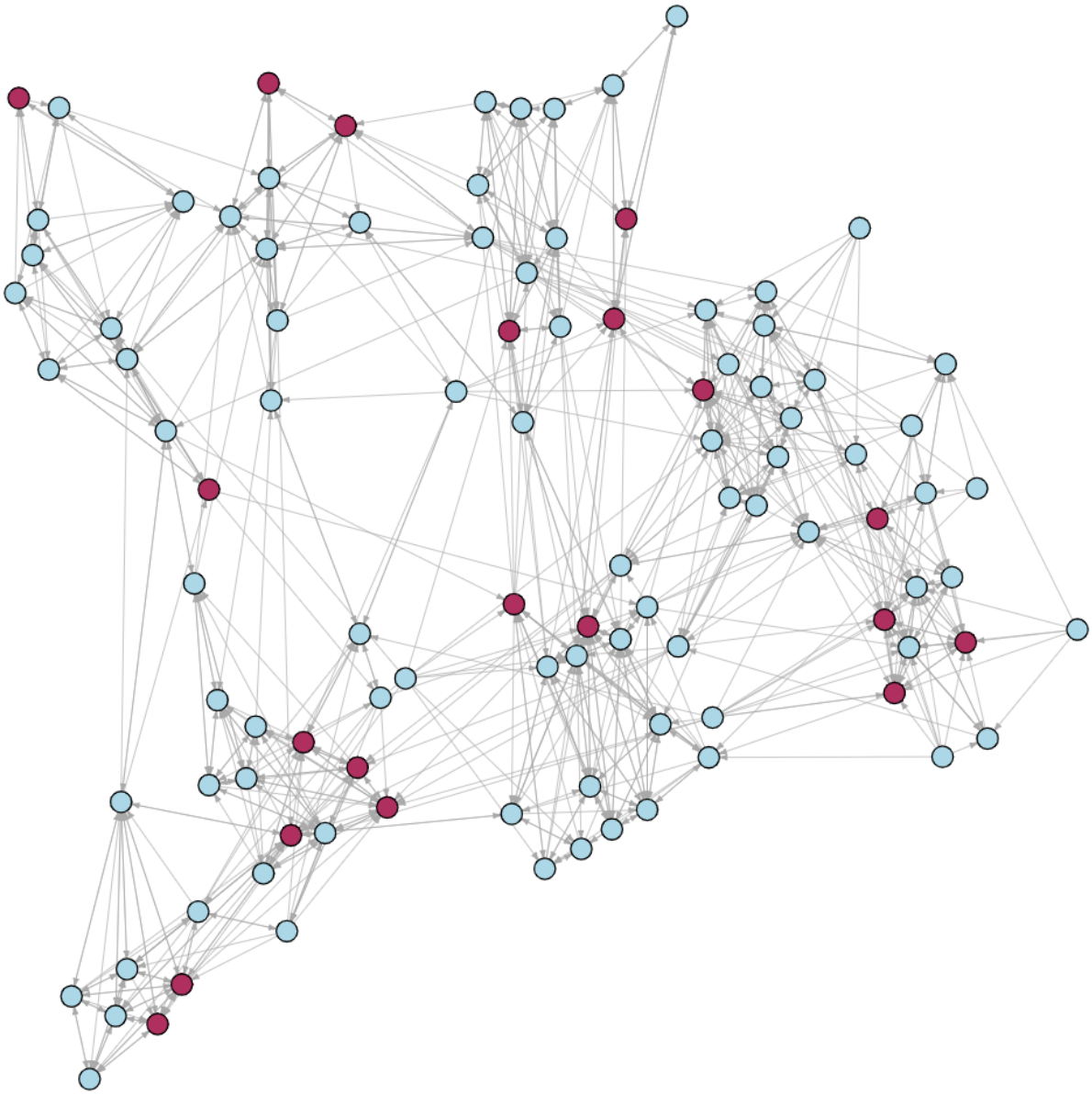


Figure 1: Spent Time Layer

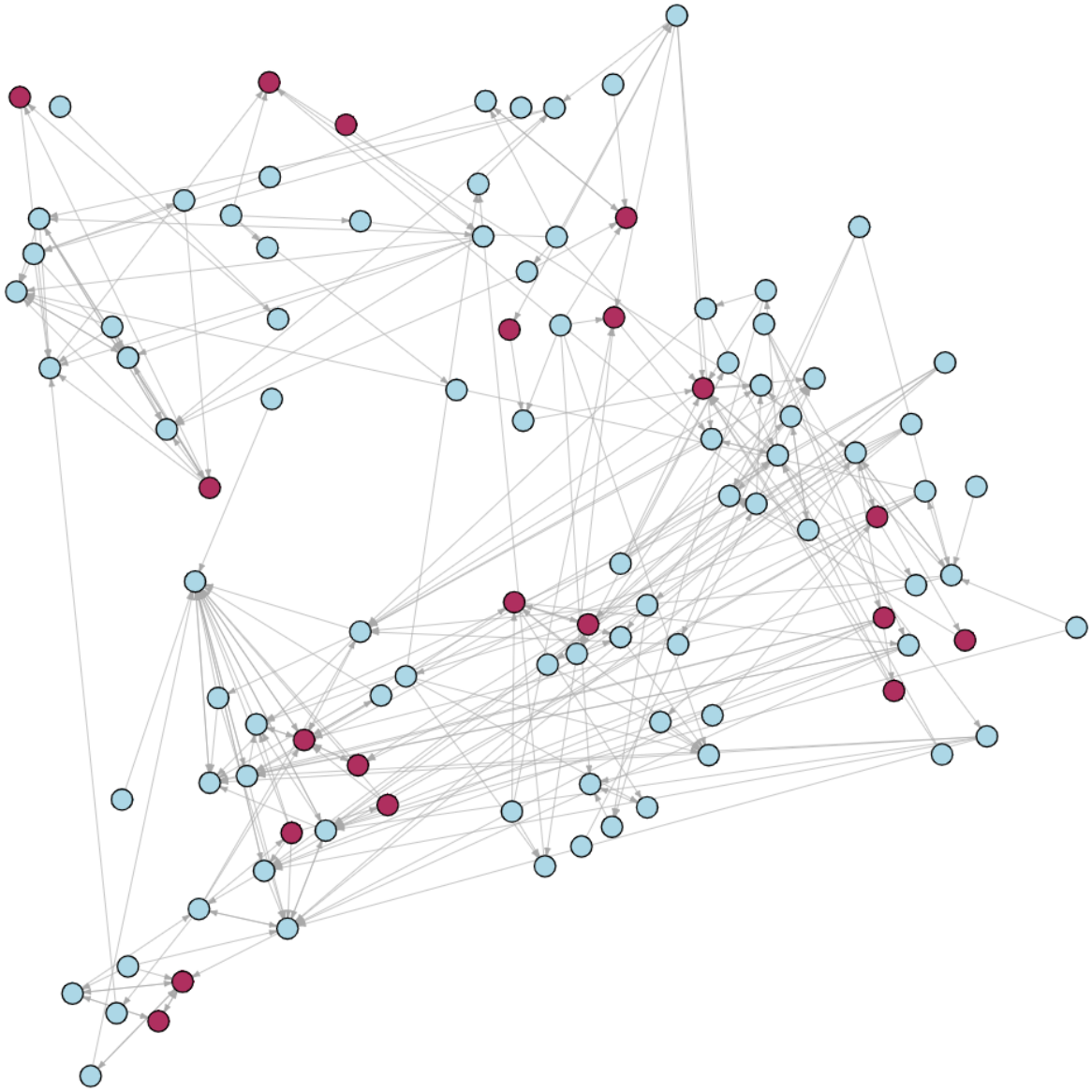


Figure 2: Conflict Layer

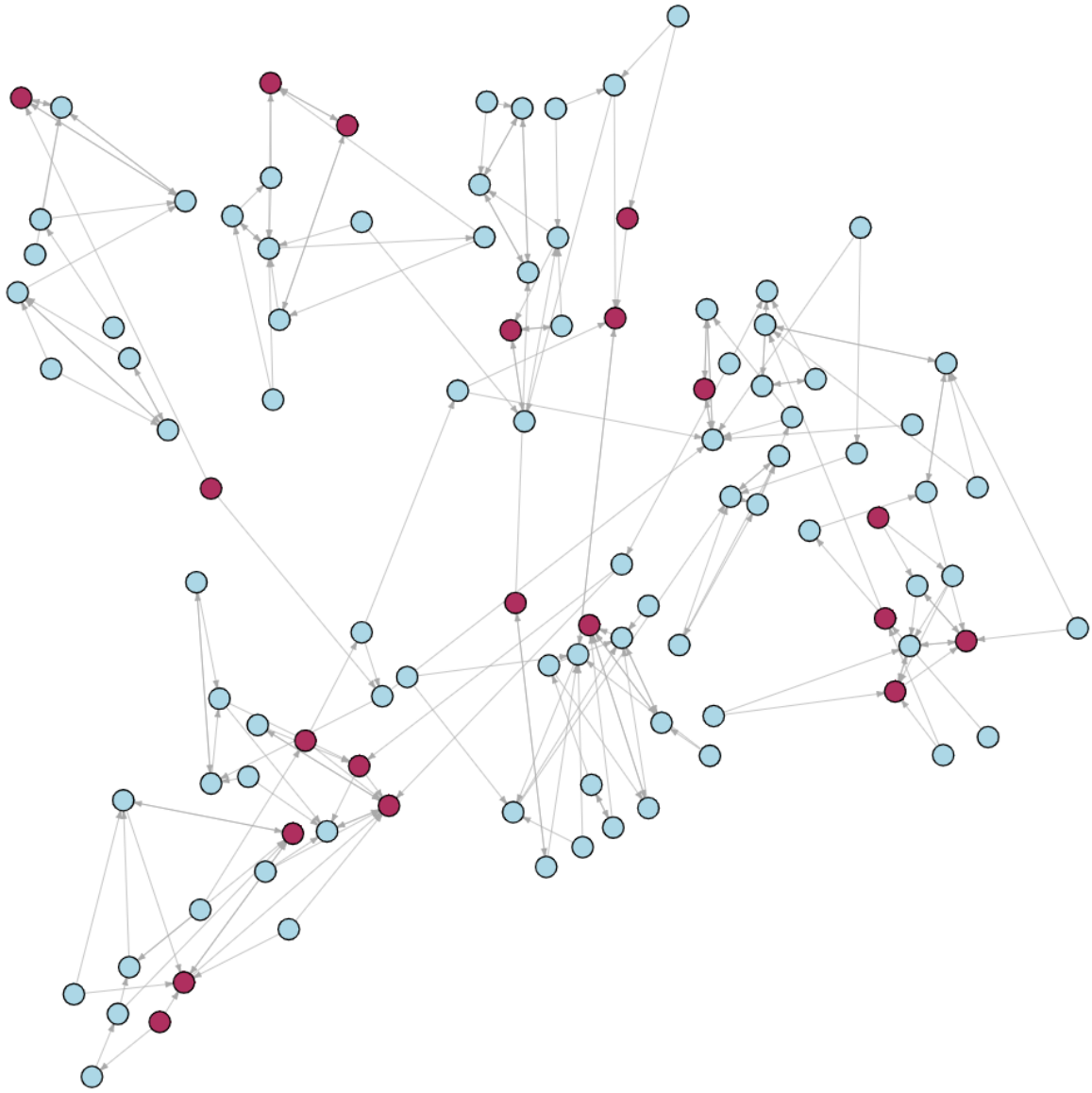


Figure 3: Best Friends Layer

process. Below, in code snippet (1), use the `intergraph` package to convert our layers, created in `igraph`, to `network` package objects. The `multilayer.ergm` package instantiates a multiplex network from the two layers. Attribute *GENC*, indicating students' gender, is added to each node and we confirm that the data structure is indeed a multilayer network.

```
spendtime_layer_n <- asNetwork(spendtime_layer)
conflict_layer_n <- asNetwork(conflict_layer)


school_mpn <- to.multiplex(
  spendtime_layer_n,
  conflict_layer_n,
  output = "network")


set.vertex.attribute(school_mpn, "GENC", rep(school_1.v$GENC,2))


check.multilayer(school_mpn)
```

Figure 4: Instantiating Multiplex Data Structure

Layer membership for all nodes are properly specified.

Next, in code snippet (2), we assign the vertex count to `vcount` and create two 103 by 103 matrices to serve as layer data structures. We specify `offzeros = TRUE` in keeping with the definition of a supra-adjacency matrix for multiplex networks. `to.multiplex` constructs the object that specifies the family of networks in ERGM's network probability distribution. This will serve as the network constraint when `multilayer.ergm` simulates potential networks in the fitting process and for network diagnostics.

```
vcount <- gorder(spendtime_layer)


sampling_constraints <- to.multiplex(
  matrix(1, ncol = vcount, nrow = vcount),
  matrix(1, ncol = vcount, nrow = vcount),
  output = "network", offzeros = TRUE)
```

Figure 5: Instantiation of Sampling Constraints

At last we specify the ERGM model in code snippet (3), using the fitting function from the `ergm` package.

`edges-layer` terms provide intercept terms for both layers, `mutual` captures reciprocal relationships in each layer, `nodeifactor_layer` measures the effect of gender on incoming ties on a given layer, and `nodeofactor_layer` measures the effect of gender on outgoing ties on a given layer. `duplexdyad` conducts a census of dyads connected on both layers. The multiplex structure is assigned as a sampling constraint along with a maximum outdegree of eleven. See `ergm` and `multilayer.ergm` packages for more information.

```
mmodel.cross <- ergm(school_mpn
                  ~ edges_layer(layer = 1)
                  + edges_layer(layer = 2)
                  + nodeifactor_layer("GENC",layer=1)
                  + nodeifactor_layer("GENC",layer=2)
                  + nodeofactor_layer("GENC",layer=1)
                  + nodeofactor_layer("GENC",layer=2)
                  + mutual("layer.mem", diff = T)
                  + duplexdyad(c("e", "f","g","h"), layers = list(1, 2)),
                  control = control.ergm(seed = 4412021),
                  constraints = ~fixallbut(sampling_constraints) + bd(maxout=11),
                  verbose=3)
```

```
summary(mmodel.cross)
```

```
## Call:
## ergm(formula = school_mpn ~ edges_layer(layer = 1) + edges_layer(layer = 2) +
##     nodeifactor_layer("GENC", layer = 1) + nodeifactor_layer("GENC",
##     layer = 2) + nodeofactor_layer("GENC", layer = 1) + nodeofactor_layer("GENC",
##     layer = 2) + mutual("layer.mem", diff = T) + duplexdyad(c("e",
##     "f", "g", "h"), layers = list(1, 2)), constraints = ~fixallbut(sampling_constraints) +
##     bd(maxout = 11), control = control.ergm(seed = 4412021),
##     verbose = 3)
##
## Monte Carlo Maximum Likelihood Results:
##
##                                  Estimate Std. Error MCMC % z value Pr(>|z|)
## edges_layer.1                   -3.396359   0.091447      0 -37.140  < 1e-04
## edges_layer.2                   -4.080846   0.125110      0 -32.618  < 1e-04
## nodeifactor_layer.1.GENC.(1) Boy -0.097287   0.092598      0  -1.051 0.293421
## nodeifactor_layer.2.GENC.(1) Boy -0.513494   0.132482      0  -3.876 0.000106
## nodeofactor_layer.1.GENC.(1) Boy  0.081233   0.114969      0   0.707 0.479840
## nodeofactor_layer.2.GENC.(1) Boy  0.246143   0.132403      0   1.859 0.063020
## mutual.same.layer.mem.1           3.777271   0.146957      0  25.703  < 1e-04
## mutual.same.layer.mem.2           1.742009   0.420557      0   4.142  < 1e-04
## duplexdyad.e                      1.493605   0.231161      0   6.461  < 1e-04
## duplexdyad.f                      1.628843   0.225100      0   7.236  < 1e-04
## duplexdyad.g                     -1.215686   0.312272      0  -3.893  < 1e-04
## duplexdyad.h                      0.006345   0.327820      0   0.019 0.984558

##      Null Deviance:     0  on 21012  degrees of freedom
##  Residual Deviance: -5978  on 21000  degrees of freedom
##
## Note that the null model likelihood and deviance are defined to be 0.
## This means that all likelihood-based inference (LRT, Analysis of
## Deviance, AIC, BIC, etc.) is only valid between models with the same
## reference distribution and constraints.
##
## AIC: -5954  BIC: -5858  (Smaller is better. MC Std. Err. = 3.124)
```

Note the model summary output of output snippets (1) and (2) above. At this point we reproduce and address our three hypotheses.

(1) Boys will be more likely than girls to be nominated as involved in a conflict relationship, all else being equal.

The coefficient associated with covariate *nodeifactor_layer.2.GENC.(1) Boy* is estimated as -0.513494 with a standard error of 0.132482. Thus the odds that a student who is nominated as involved in a conflict relationship if they are a boy is 0.5984011 times the odds of the same nomination if the student were a girl, all else being equal. As the associated p-value is 0.000106, at the $\alpha = 0.05$ level we reject the null hypothesis that this effect is not statistically significant (equal to one). Since the odds of nomination are higher for girls than boys, our hypothesis is disproved.

(2) If a student nominates a peer as spending time with them, the nominated person will be more likely to nominate the first student than if they were not nominated, all else being equal.

The coefficient associated with covariate *mutual.same.layer.mem.1* is estimated as 3.777271 with a standard error of 0.146957. Thus the odds that student u nominated student v as someone they spend time with if v nominated u as someone they spend time with are 43.69663 times the odds if v did not nominate u, all else being equal. As the associated p-value is less than 0.001, at the $\alpha = 0.05$ level we reject the null hypothesis that this effect is not statistically significant (equal to one). Since the odds of nomination of v are dramatically higher if v nominated u than if they did not, our hypothesis is affirmed.

(3) If two students nominate each other as involved in a friendship relationship, it will be less likely that one of those students will nominate the other as involved in a conflict relationship than if the two students did not nominate each other as involved in a friendship relationship, all else being equal.

The coefficient associated with covariate *duplexdyad.g* is estimated as -1.215686 with a standard error of 0.312272. Thus the odds that one student nominates another as involved in a conflict relationship with them given that they are nominated each other as someone they spend time with is 0.2965065 times the odds of the conflict nomination if there was not a mutual nomination of spending time together, all else being equal. As the associated p-value is less than 0.001, at the $\alpha = 0.05$ level we reject the null hypothesis that this effect is not statistically significant (equal to one). Since the odds of a single conflict nomination are much lower if there was a mutual nomination of time spent together compared to no such mutual nomination, our third hypothesis is supported.

Overall, two of our hypotheses were backed up by the data, and a third was disproven. As always, it is essential

to validate the soundness of point estimates with overall model diagnostics. Inspection of trace plots and histograms relating to the MCMC process proceeds as in the standard ERGM case, with `mcmc.diagnostics()` in code snippet (5).

```
mcmc.diagnostics(mmodel.cross)
```

Figure 6: MCMC Diagnostics Call

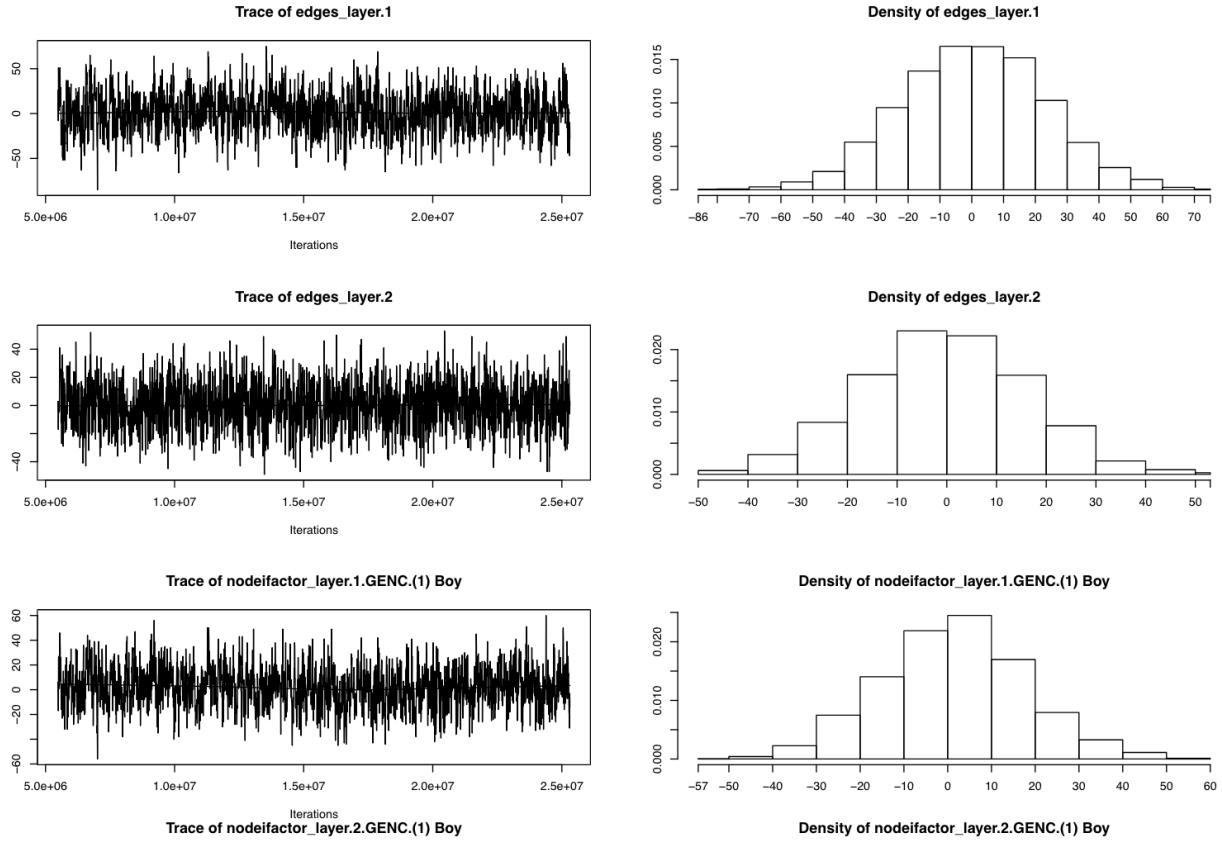


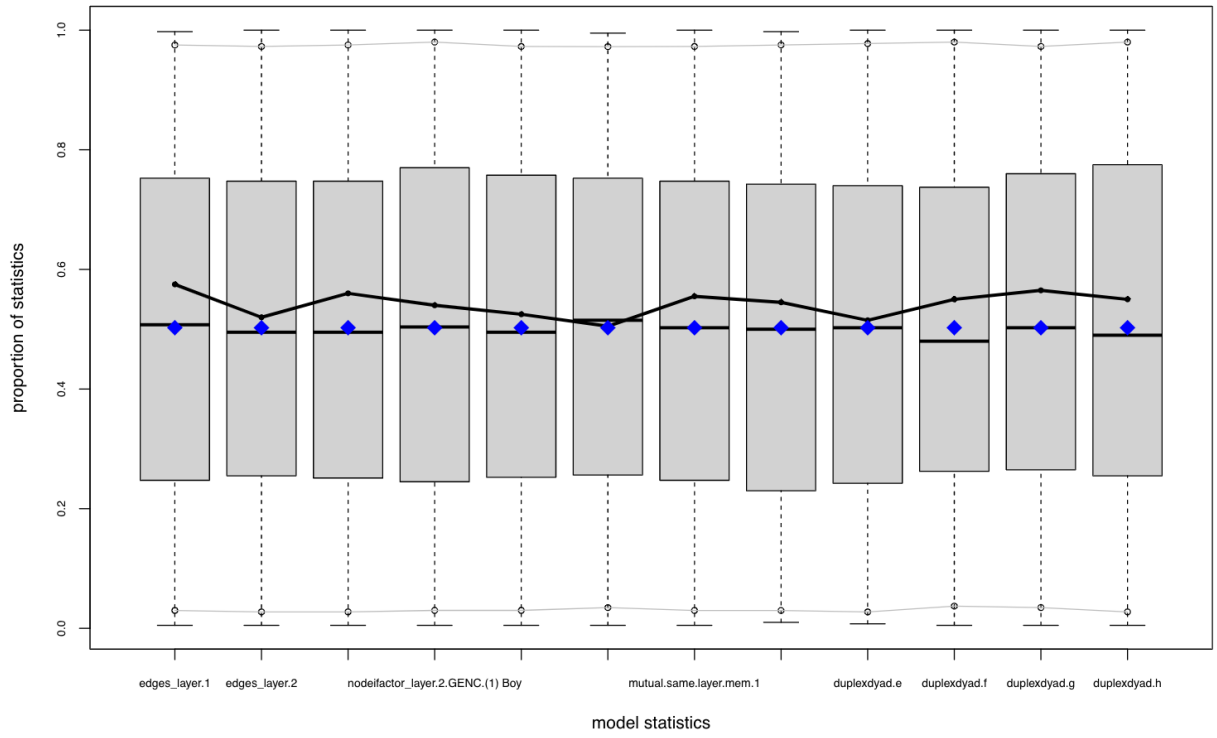
Figure 7: MCMC Diagnostics Example

See above an example of MCMC diagnostics, figure (4). The trace plot reveals no significantly concerning trends, and the variance appears acceptable. Histograms seem normal in distribution. MCMC diagnostics for the remaining estimated coefficients appear similar, indicating that the algorithm converged successfully.

Next we plot and output goodness-of-fit diagnostics, based on the `gof` method from the standard `ergm` package. A sample of the results follow in figure (5) and code output (3), (4).

```
gof <- gof(mmodel.cross, verbose=T, burnin=1e+7, interval=1e+5, control = control.gof.ergm(nsim = 200))
plot(gof)
```

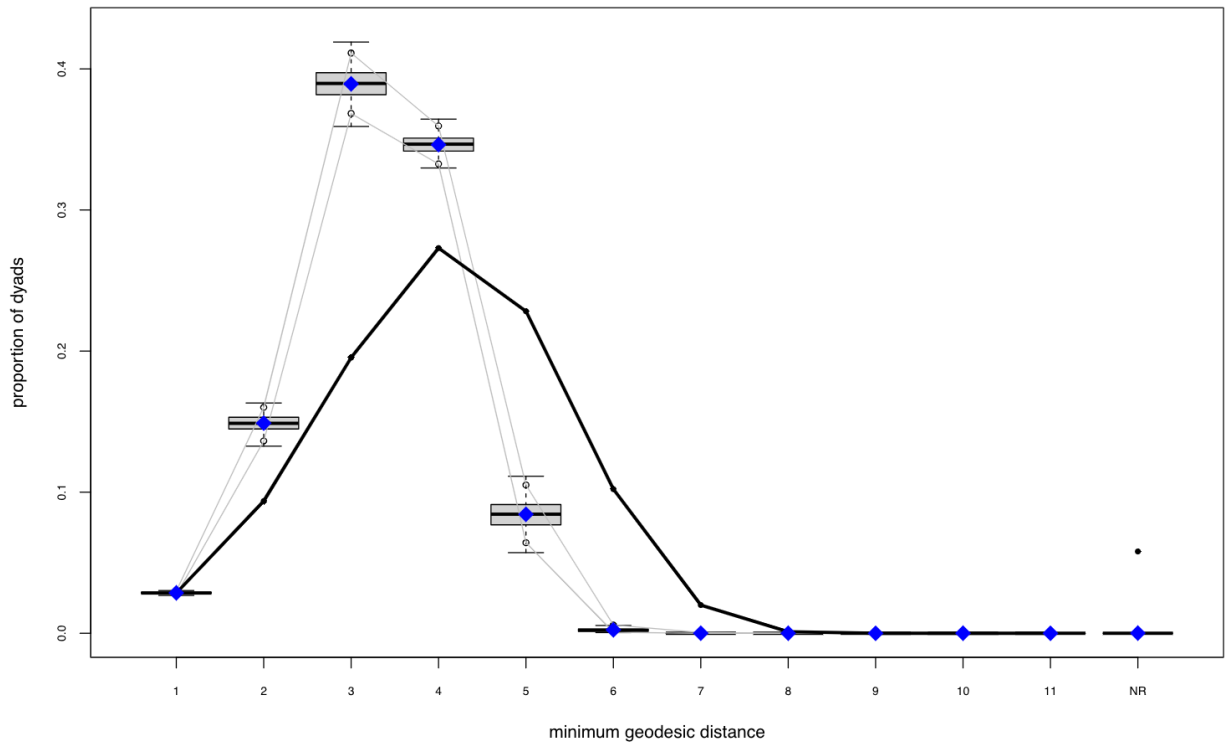
Figure 8: Goodness-of-fit Plot Call




```
## Goodness-of-fit for model statistics
##
##                                        obs min    mean max MC p-value
```

```
## edges_layer.1                        738 684 742.530 793       0.88
## edges_layer.2                        259 211 259.570 306       1.00
## nodeifactor_layer.1.GENC.(1) Boy     374 331 376.070 422       0.92
## nodeifactor_layer.2.GENC.(1) Boy     106  83 106.515 136       1.00
## nodeofactor_layer.1.GENC.(1) Boy     388 338 388.505 427       1.00
## nodeofactor_layer.2.GENC.(1) Boy     146 110 145.765 172       1.00
## mutual.same.layer.mem.1              213 188 214.820 242       0.98
## mutual.same.layer.mem.2               23  12  22.950  35       1.00
## duplexdyad.e                          80  60  80.515 108       1.00
## duplexdyad.f                          82  61  82.115 110       1.00
## duplexdyad.g                          54  35  54.135  76       0.94
## duplexdyad.h                          27  11  26.730  44       1.00
```

The model statistics versus proportion of statistics plot indicates that the fitted model is a good fit for each parameter in the multiplex ERGM model. Our observed network's statistics, illustrated by the thick black line, falls within the interquartile range of the simulated networks' corresponding statistics, which are illustrated by the boxplots. In addition, the p-values associated with the statistics' goodness-of-fit do not fall below 0.88. At the $\alpha = 0.05$ level, we can reject the null hypothesis of a poor model fit for each of the statistics.


proportion of dyads
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
7
8
9
10
11
NR
minimum geodesic distance

```
## Goodness-of-fit for minimum geodesic distance
##
##        obs   min      mean   max MC p-value
## 1     1203  1131  1208.100  1305       0.93
## 2     3956  5600  6290.820  7093       0.00
## 3     8250 15169 16442.410 18002       0.00
## 4    11529 13578 14622.500 15389       0.00
## 5     9638  2230  3558.980  4800       0.00
## 6     4314    20   101.295   285       0.00
## 7      845     0     1.780    74       0.00
## 8       47     0     0.035     4       0.00
## Inf   2448     0     4.080   408       0.00
```

Notice also the geodesic distance versus proportion of dyads plot, above in figure (6) and code output (5). Clearly, the model is a poor fit in terms of minimum geodesic distance given that the observed proportion of dyads for the majority of geodesic distances falls outside of their corresponding simulated results' interquartile ranges. The associated p-values confirm this insight, as the majority fall below 0.01. So at the $\alpha = 0.05$ level, for most levels of minimum geodesic distance we fail to reject the null hypothesis that the model is a poor fit in terms of those minimum geodesic distances.

In all likelihood, the model is indeed a poor fit in terms of minimum geodesic distance, edge-wise shared partners, indegree, and outdegree. More model terms are needed to successfully model our observed network and our earlier conclusions are cast in doubt. As indicated earlier, however, these goodness-of-fit diagnostics fail to take the layered structure of our network into account. Ideally we would be inspecting separate plots for each layer as well as additional diagnostics based on cross-layer statistics. Thus the implementation of ERGM in R still suffers from a critical gap in relevant software packages.

Conclusion

Using data on the interrelationships of schoolchildren, we constructed and visualized a multiplex network with three distinct layers. We fit an ERGM model on two of those layers to conduct statistical inference, and gauged the performance of the overall model. This demonstration builds upon our earlier theoretical exploration of the multiplex network structure.

Initially we used the multilayer network framework introduced by (Kivela et. al., 2014) to contextualize, define, and represent these novel network types. Then a high-level showcase of network descriptors and motifs illustrated how multiplex networks can be described, analyzed, and differentiated. We shifted the expression and methodology behind exponential random graph models to address the multiplicity of layers. Ultimately we reviewed a number of R packages that work with multilayer networks and concluded with the preceding application study. Important extensions to the techniques discussed here include the implementation of goodness-of-fit measures for multiplex ERGMs, algorithms optimized for the fitting of said ERGMs, additional network motifs in the two-layer setting, and extensions to allow for more layers and aspects.

With this background in multiplex network theory and implementation, the reader has a solid toolbox for reviewing the relevant literature and undertaking their own analysis.

Works Cited

Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., & Pedreschi, D. (2011). Foundations of Multidimensional Network Analysis. 2011 International Conference on Advances in Social Networks Analysis and Mining, 485–489. https://doi.org/10.1109/ASONAM.2011.103

Berlingerio, M., Coscia, M., Giannotti, F., Monreale, A., & Pedreschi, D. (2013). Multidimensional networks: foundations of structural analysis. World Wide Web, 16(5), 567–593. https://doi.org/10.1007/s11280-012-0190-4

Besag. (1975). Statistical Analysis of Non-Lattice Data. Journal of the Royal Statistical Society. Series D (The Statistician), 24(3), 179–195. https://doi.org/10.2307/2987782

Chatterjee, & Diaconis, P. (2013). ESTIMATING AND UNDERSTANDING EXPONENTIAL RANDOM GRAPH MODELS. The Annals of Statistics, 41(5), 2428–2461. https://doi.org/10.1214/13-AOS1155

Chen, T. H. (2019). Statistical inference for multilayer networks in political science. Political Science Research and Methods, 9(2), 380–397. https://doi.org/10.1017/psrm.2019.49

Chen, T. H. (2019). Tedhchen/multilayer.ergm: R package for fitting ergms for multilayer networks. GitHub. Retrieved October 22, 2021, from https://github.com/tedhchen/multilayer.ergm.

Csárdi, G. (2020). Igraph R package. https://igraph.org/r/

De Domenico, Sole-Ribalta, A., Cozzo, E., Kivela, M., Moreno, Y., Porter, M. A., Gomez, S., & Arenas, A. (2013). Mathematical Formulation of Multilayer Networks. Physical Review. X, 3(4), 041022–. https://doi.org/10.1103/PhysRevX.3.041022

De Domenico, M., Porter, M. A., & Arenas, A. (2015). MuxViz: a tool for multilayer analysis and visualization of networks. Journal of Complex Networks, 3(2), 159–176. https://doi.org/10.1093/comnet/cnu03

Frank, & Strauss, D. (1986). Markov Graphs. Journal of the American Statistical Association, 81(395), 832–842. https://doi.org/10.1080/01621459.1986.10478342

Ghafouri, & Khasteh, S. H. (2020). A survey on exponential random graph models: an application perspective. PeerJ. Computer Science, 6, e269–e269. https://doi.org/10.7717/peerj-cs.269

Kivela, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y., & Porter, M. A. (2014). Multilayer networks. Journal of Complex Networks, 2(3), 203–271. https://doi.org/10.1093/comnet/cnu016

Lusher, D., Koskinen, J., & Robins, G. (Eds.). (2012). Exponential Random Graph Models for Social Networks: Theory, Methods, and Applications (Structural Analysis in the Social Sciences). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511894701

Magnani, M., Rossi, L., & Vega, D. (2021). Analysis of Multiplex Social Networks with R. Journal of Statistical Software, 98(8), 1–. https://doi.org/10.18637/jss.v098.i08

Morris, Handcock, M. S., & Hunter, D. R. (2008). Specification of Exponential-Family Random Graph Models: Terms and Computational Aspects. Journal of Statistical Software, 24(4), 1548–7660. https://doi.org/10.18637/jss.v024.i04

Nordhaus, E., & Stewart, B. (1963). Triangles in an Ordinary Graph. Canadian Journal of Mathematics, 15, 33-41. doi:10.4153/CJM-1963-004-7

Paluck, E. L., Shepherd, H., & Aronow, P. M. (2016). Changing climates of conflict: A social network experiment in 56 schools. Proceedings of the National Academy of Sciences - PNAS, 113(3), 566–571. https://doi.org/10.1073/pnas.1514483113

Plotly Technologies Inc. (2015). Collaborative data science. https://plot.ly

Tyner, Briatte, F., & Hofmann, H. (2017). Network Visualization with ggplot2. The R Journal, 9(1), 27–. https://doi.org/10.32614/RJ-2017-023

Wang, Robins, G., Pattison, P., & Lazega, E. (2013). Exponential random graph models for multilevel networks. Social Networks, 35(1), 96–115. https://doi.org/10.1016/j.socnet.2013.01.004

Appendix I: Application Code

Pages with the example application's code follow. Code was written in a RMarkdown file with RStudio, then exported to PDF form.

Tutorial: Multiplex Networks

Tyler Maule

November 28th, 2021

```
#import packages needed for this application
library(igraph)
library(tidyverse)
library(multilayer.ergm)
library(multinet)
library(muxViz)
library(intergraph)
library(reshape)
library(rio)
```

```
#load and import data from Paluck et. al.
#(accessible at https://www.icpsr.umich.edu/web/civicleads/studies/37070/publications)
school_df<- import("ICPSR_37070/DS0001/37070-002-Data.csv")


set.seed(441)


#select school ID no. 15
school_1 <- school_df %>% filter((SCHID == 15))
```

```
#create a list of vertices (nodes) from the list of students at school 15,
#along with key characteristics of each student
school_1.v <- school_1 %>% select(ID, TREAT, GENC)


#minor data processing -- convert the treatment variable into a binary indicator
school_1.v$TREAT2 <- ifelse(school_1.v$TREAT == "(1) Treatment", 1, 0)
```

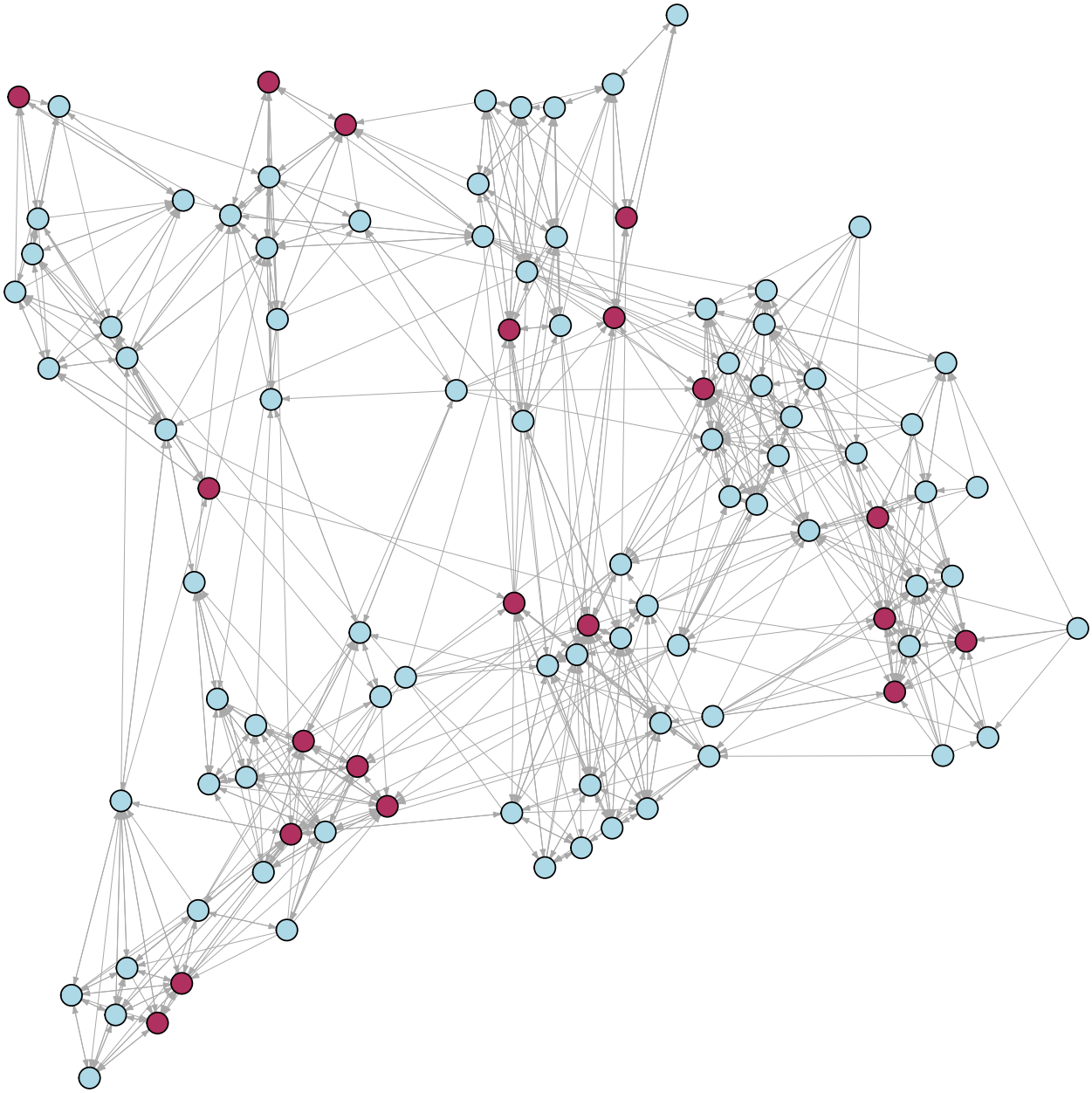
```r
school_1.v$ID <- as.numeric(school_1.v$ID)


school_1.v[1:3,]
```

```
##   ID                          TREAT     GENC TREAT2
## 1  1                  (2) Control (0) Girl      0
## 2  2 (0) Not treatment or control (0) Girl      0
## 3  3 (0) Not treatment or control (0) Girl      0
```

```r
spendtime_noms <- c("ID","ST1", "ST2", "ST3", "ST4", "ST5", "ST6", "ST7", "ST8", "ST9", "ST10")
conflict_noms <- c("ID","CN1", "CN2", "CN3", "CN4", "CN5")
bestfriend_noms <- c("ID","BF1", "BF2")


get_layer_edges <- function(layer_nominations){
  #create a dataframe with two columns, where each row will represent an edge:
  #the first capturing each student ID, and the second capturing IDs of students
  #who were identified as friends OR best friends, OR peers with whom the
  #first student has conflict
  school_1.e <- school_1[,layer_nominations]
  school_1.e <- melt(school_1.e, id.vars="ID", var="NOM_TYPE")
  colnames(school_1.e) <- c("ID","NOM_TYPE","NEIGHBOR")
  school_1.e <- school_1.e[,c("ID","NEIGHBOR")]


  school_1.e$ID <- as.numeric(school_1.e$ID)
  school_1.e$NEIGHBOR <- as.numeric(school_1.e$NEIGHBOR)


  #ensure that there are no duplicate edges or edges not in the vertex list
  school_1.e <- school_1.e[school_1.e$NEIGHBOR %in% school_1.v$ID,]


  school_1.e
}


get_layer_igraph <- function(layer_edgelist){
```

```
  #use the igraph package to create a graph object based on the node dataframe
  #and edge dataframe; simplify the graph to remove any remaining multi-edges and loops
  school_1.g = graph_from_data_frame(layer_edgelist, directed = TRUE, vertices = school_1.v)
  school_1.g = igraph::simplify(school_1.g)

  school_1.g
}

visualize_layer_igraph <- function(layer_igraph, layout_pref){
  #adjust visualization settings to color directly treated nodes in red,
  #and the control nodes in blue
  V(layer_igraph)$color <- ifelse(V(layer_igraph)$TREAT2 == 1, "maroon", "light blue")
  igraph_options(vertex.size = 4, vertex.label = NA, edge.arrow.size = 0.25, edge.width = 0.5)

  #plot by letting igraph choose a "nice" layout based on the graph structure
  par(mar=c(0,0,0,0))
  plot(layer_igraph, layout = layout_pref)
}
```

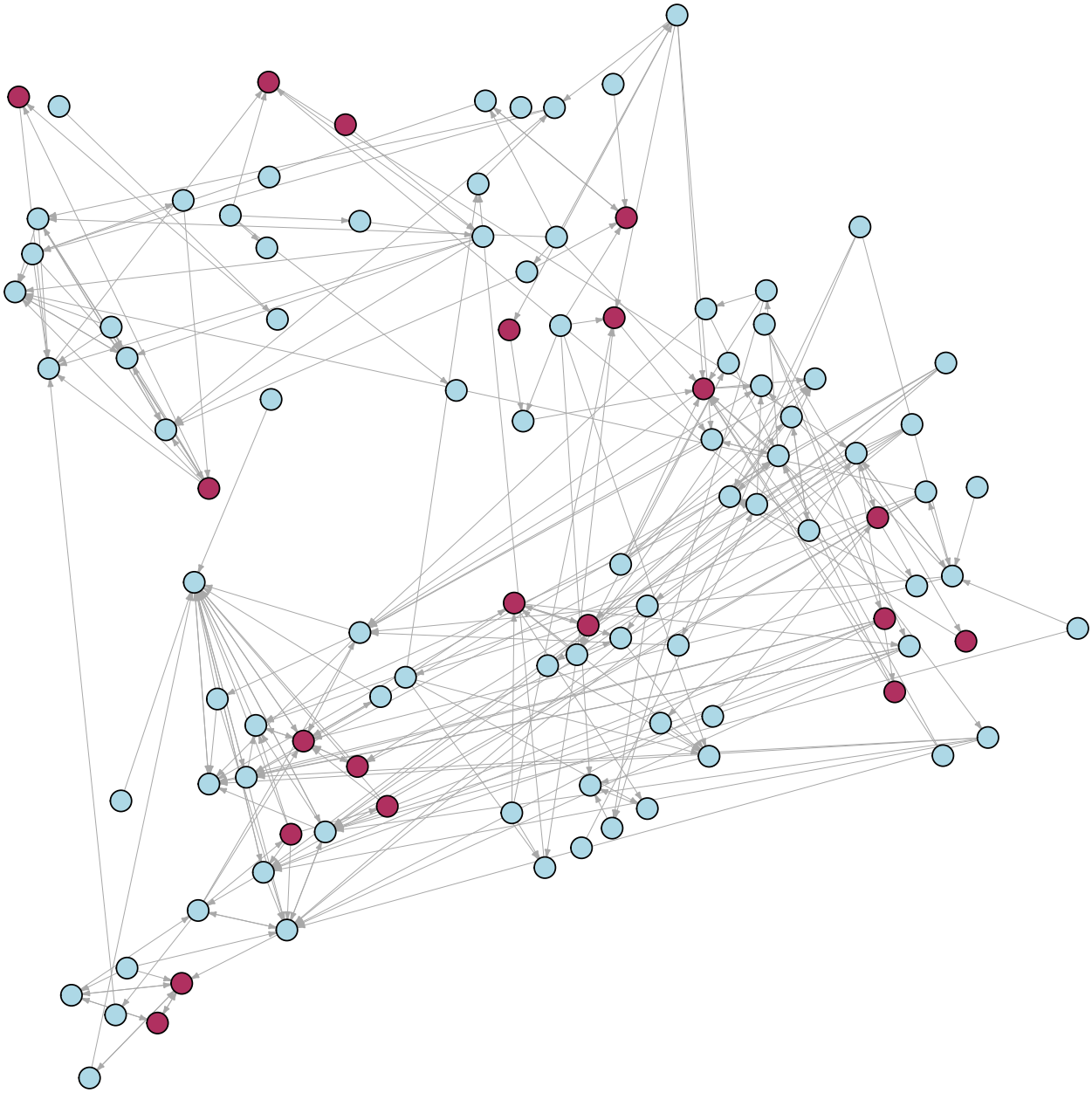
```
#create and plot monoplex "spendtime" network (layer I)
spendtime_edges <- get_layer_edges(spendtime_noms)
spendtime_layer <- get_layer_igraph(spendtime_edges)

school_layout <- layout.fruchterman.reingold(spendtime_layer)
visualize_layer_igraph(spendtime_layer,layout_pref=school_layout)
```



```
summary(spendtime_layer)
```

```
## IGRAPH e6e9dd2 DN-- 103 738 --
## + attr: name (v/c), TREAT (v/c), GENC (v/c), TREAT2 (v/n)
```

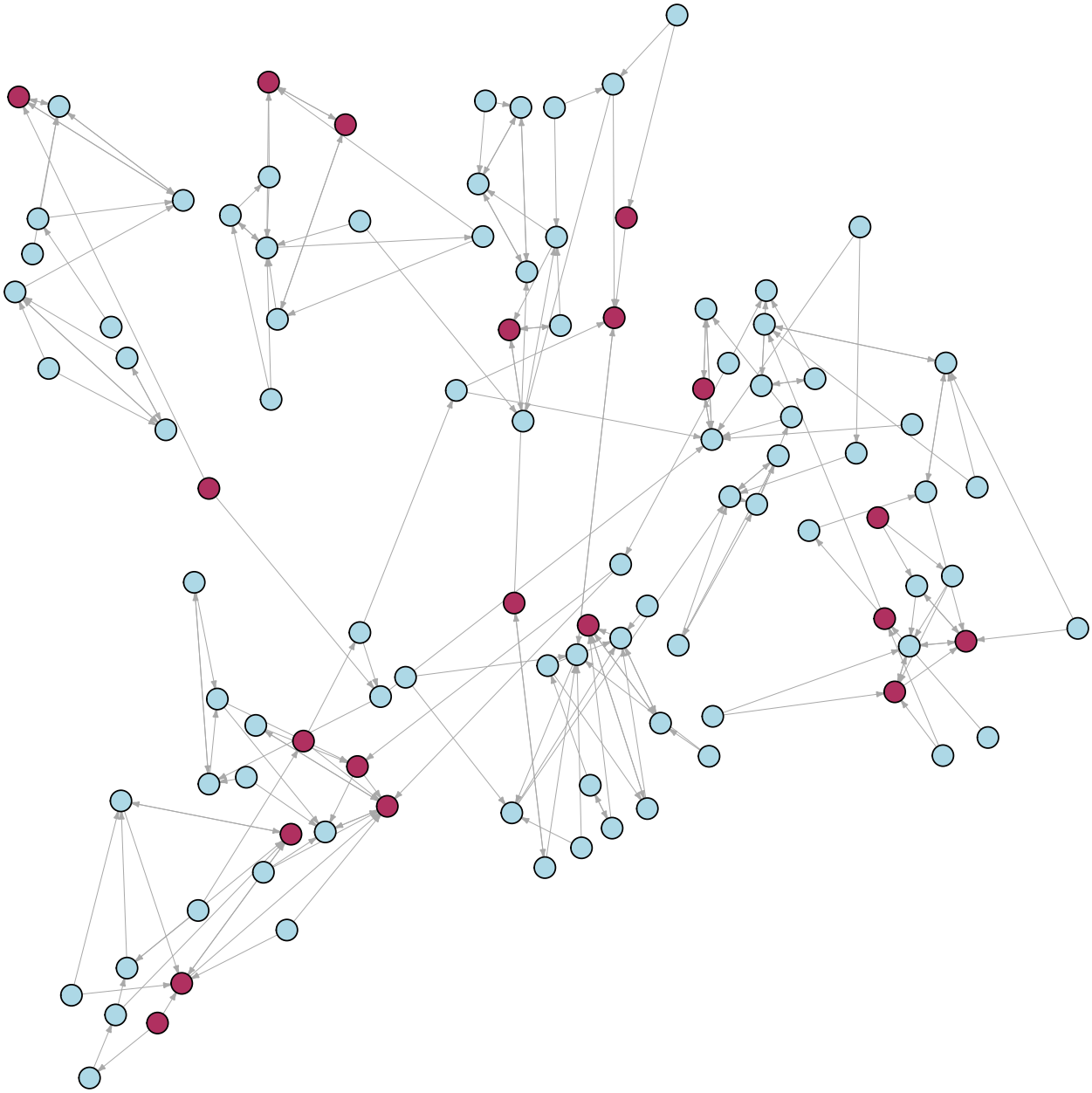
```
#create and plot monoplex "conflict" network (layer II)
conflict_edges <- get_layer_edges(conflict_noms)
conflict_layer <- get_layer_igraph(conflict_edges)
visualize_layer_igraph(conflict_layer,layout_pref=school_layout)
```



```
summary(conflict_layer)
```

```
## IGRAPH eee1413 DN-- 103 259 --
## + attr: name (v/c), TREAT (v/c), GENC (v/c), TREAT2 (v/n)
```

```
#create and plot monoplex "bestfriend" network (layer III)
bestfriend_edges <- get_layer_edges(bestfriend_noms)
bestfriend_layer <- get_layer_igraph(bestfriend_edges)
visualize_layer_igraph(bestfriend_layer,layout_pref=school_layout)
```



```
summary(bestfriend_layer)
```

```
## IGRAPH eacc66d DN-- 103 193 --
## + attr: name (v/c), TREAT (v/c), GENC (v/c), TREAT2 (v/n)
```

```
spendtime_layer_n <- asNetwork(spendtime_layer)
conflict_layer_n <- asNetwork(conflict_layer)


school_mpn <- to.multiplex(
```

```
  spendtime_layer_n,
  conflict_layer_n,
  output = "network")


set.vertex.attribute(school_mpn, "GENC", rep(school_1.v$GENC,2))


check.multilayer(school_mpn)
```

```
(school_mpn)
```

```
##  Network attributes:
##   vertices = 206
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 1203
##     missing edges= 0
##     non-missing edges= 1203
##
##  Vertex attribute names:
##     GENC layer.mem vertex.names
##
##  Edge attribute names not shown
```

```
vcount <- gorder(spendtime_layer)


sampling_constraints <- to.multiplex(
  matrix(1, ncol = vcount, nrow = vcount),
  matrix(1, ncol = vcount, nrow = vcount),
  output = "network", offzeros = TRUE)
```

```
mmodel.cross <- ergm(school_mpn
                  ~ edges_layer(layer = 1)
                  + edges_layer(layer = 2)
                  + nodeifactor_layer("GENC",layer=1)
                  + nodeifactor_layer("GENC",layer=2)
                  + nodeofactor_layer("GENC",layer=1)
                  + nodeofactor_layer("GENC",layer=2)
                  + mutual("layer.mem", diff = T)
                  + duplexdyad(c("e", "f","g","h"), layers = list(1, 2)),
                  control = control.ergm(seed = 4412021),
                  constraints = ~fixallbut(sampling_constraints) + bd(maxout=11),
                  verbose=0)

summary(mmodel.cross)
```

```
## Call:
## ergm(formula = school_mpn ~ edges_layer(layer = 1) + edges_layer(layer = 2) +
##     nodeifactor_layer("GENC", layer = 1) + nodeifactor_layer("GENC",
##     layer = 2) + nodeofactor_layer("GENC", layer = 1) + nodeofactor_layer("GENC",
##     layer = 2) + mutual("layer.mem", diff = T) + duplexdyad(c("e",
##     "f", "g", "h"), layers = list(1, 2)), constraints = ~fixallbut(sampling_constraints) +
##     bd(maxout = 11), control = control.ergm(seed = 4412021),
##     verbose = 0)
##
## Monte Carlo Maximum Likelihood Results:
##
##                                  Estimate Std. Error MCMC % z value Pr(>|z|)
## edges_layer.1                   -3.396359   0.091447      0 -37.140  < 1e-04
## edges_layer.2                   -4.080846   0.125110      0 -32.618  < 1e-04
## nodeifactor_layer.1.GENC.(1) Boy -0.097287   0.092598      0  -1.051 0.293421
## nodeifactor_layer.2.GENC.(1) Boy -0.513494   0.132482      0  -3.876 0.000106
## nodeofactor_layer.1.GENC.(1) Boy  0.081233   0.114969      0   0.707 0.479840
## nodeofactor_layer.2.GENC.(1) Boy  0.246143   0.132403      0   1.859 0.063020
```

```
## mutual.same.layer.mem.1           3.777271   0.146957      0  25.703  < 1e-04
## mutual.same.layer.mem.2           1.742009   0.420557      0   4.142  < 1e-04
## duplexdyad.e                      1.493605   0.231161      0   6.461  < 1e-04
## duplexdyad.f                      1.628843   0.225100      0   7.236  < 1e-04
## duplexdyad.g                     -1.215686   0.312272      0  -3.893  < 1e-04
## duplexdyad.h                      0.006345   0.327820      0   0.019 0.984558
##
## edges_layer.1                    ***
## edges_layer.2                    ***
## nodeifactor_layer.1.GENC.(1) Boy
## nodeifactor_layer.2.GENC.(1) Boy ***
## nodeofactor_layer.1.GENC.(1) Boy
## nodeofactor_layer.2.GENC.(1) Boy .
## mutual.same.layer.mem.1          ***
## mutual.same.layer.mem.2          ***
## duplexdyad.e                     ***
## duplexdyad.f                     ***
## duplexdyad.g                     ***
## duplexdyad.h
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance:     0  on 21012  degrees of freedom
##  Residual Deviance: -5978  on 21000  degrees of freedom
##
## Note that the null model likelihood and deviance are defined to be 0.
## This means that all likelihood-based inference (LRT, Analysis of
## Deviance, AIC, BIC, etc.) is only valid between models with the same
## reference distribution and constraints.
##
## AIC: -5954  BIC: -5858  (Smaller is better. MC Std. Err. = 3.124)
```

```
mcmc.diagnostics(mmodel.cross)
```

```
## Sample statistics summary:
##
## Iterations = 5480448:25313280
## Thinning interval = 8192
## Number of chains = 1
## Sample size per chain = 2422
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                                      Mean     SD Naive SE Time-series SE
## edges_layer.1                     1.23121 22.553  0.45826         0.8614
## edges_layer.2                     0.10859 16.394  0.33311         0.4405
## nodeifactor_layer.1.GENC.(1) Boy 1.94220 16.442  0.33410         0.6371
## nodeifactor_layer.2.GENC.(1) Boy 0.36664 10.318  0.20966         0.2556
## nodeofactor_layer.1.GENC.(1) Boy 1.85797 15.102  0.30686         0.5548
## nodeofactor_layer.2.GENC.(1) Boy 0.45417 12.045  0.24475         0.3024
## mutual.same.layer.mem.1          1.38604 11.225  0.22808         0.4911
## mutual.same.layer.mem.2          0.00289  4.739  0.09629         0.1438
## duplexdyad.e                     0.20809  9.608  0.19523         0.3223
## duplexdyad.f                     0.46367  9.780  0.19873         0.3330
## duplexdyad.g                     0.49959  8.361  0.16989         0.3083
## duplexdyad.h                     0.12097  6.765  0.13746         0.2306
##
## 2. Quantiles for each variable:
##
##                                    2.5% 25% 50% 75%  97.5%
## edges_layer.1                   -43.00 -14   1  17 45.475
## edges_layer.2                   -32.00 -11   0  11 32.000
## nodeifactor_layer.1.GENC.(1) Boy -30.00  -9   2  12 35.000
## nodeifactor_layer.2.GENC.(1) Boy -19.00  -7   0   7 21.475
```

```
## nodeofactor_layer.1.GENC.(1) Boy -28.00  -8   2  12 31.000
## nodeofactor_layer.2.GENC.(1) Boy -23.47  -8   1   9 24.000
## mutual.same.layer.mem.1          -20.47  -6   1   9 23.000
## mutual.same.layer.mem.2           -9.00  -3   0   3  9.475
## duplexdyad.e                     -17.00  -6   0   7 20.000
## duplexdyad.f                     -18.00  -6   0   7 21.000
## duplexdyad.g                     -16.00  -5   0   6 17.000
## duplexdyad.h                     -12.00  -5   0   4 14.000
##
##
## Are sample statistics significantly different from observed?
##            edges_layer.1 edges_layer.2 nodeifactor_layer.1.GENC.(1) Boy
## diff.          1.2312139     0.1085879                      1.942196532
## test stat.     1.4293883     0.2464902                      3.048627643
## P-val.         0.1528927     0.8053028                      0.002298892
##            nodeifactor_layer.2.GENC.(1) Boy nodeofactor_layer.1.GENC.(1) Boy
## diff.                             0.3666391                      1.8579686210
## test stat.                        1.4343695                      3.3489431485
## P-val.                            0.1514669                      0.0008112044
##            nodeofactor_layer.2.GENC.(1) Boy mutual.same.layer.mem.1
## diff.                             0.4541701             1.386044591
## test stat.                        1.5020548             2.822546630
## P-val.                            0.1330830             0.004764389
##            mutual.same.layer.mem.2 duplexdyad.e duplexdyad.f duplexdyad.g
## diff.                  0.002890173    0.2080925    0.4636664    0.4995871
## test stat.             0.020094863    0.6457441    1.3923807    1.6204892
## P-val.                 0.983967698    0.5184451    0.1638071    0.1051272
##            duplexdyad.h Overall (Chi^2)
## diff.         0.1209744              NA
## test stat.    0.5246630    4.228834e+01
## P-val.        0.5998175    3.904744e-05
##
## Sample statistics cross-correlations:
```

```
##                                 edges_layer.1 edges_layer.2
## edges_layer.1                      1.00000000    0.15613925
## edges_layer.2                      0.15613925    1.00000000
## nodeifactor_layer.1.GENC.(1) Boy   0.67352731    0.08046948
## nodeifactor_layer.2.GENC.(1) Boy   0.08004557    0.66751616
## nodeofactor_layer.1.GENC.(1) Boy   0.76499612    0.10169272
## nodeofactor_layer.2.GENC.(1) Boy   0.11887291    0.76186879
## mutual.same.layer.mem.1            0.74497510    0.10622785
## mutual.same.layer.mem.2            0.11247989    0.53884653
## duplexdyad.e                       0.27772882    0.58797641
## duplexdyad.f                       0.26721056    0.58159304
## duplexdyad.g                       0.25946511    0.46073184
## duplexdyad.h                       0.14625026    0.44553698
##                                 nodeifactor_layer.1.GENC.(1) Boy
## edges_layer.1                                         0.67352731
## edges_layer.2                                         0.08046948
## nodeifactor_layer.1.GENC.(1) Boy                      1.00000000
## nodeifactor_layer.2.GENC.(1) Boy                      0.09063442
## nodeofactor_layer.1.GENC.(1) Boy                      0.71984280
## nodeofactor_layer.2.GENC.(1) Boy                      0.11991811
## mutual.same.layer.mem.1                               0.51790216
## mutual.same.layer.mem.2                               0.06137669
## duplexdyad.e                                          0.13810445
## duplexdyad.f                                          0.18073536
## duplexdyad.g                                          0.16263221
## duplexdyad.h                                          0.06713297
##                                 nodeifactor_layer.2.GENC.(1) Boy
## edges_layer.1                                         0.08004557
## edges_layer.2                                         0.66751616
## nodeifactor_layer.1.GENC.(1) Boy                      0.09063442
## nodeifactor_layer.2.GENC.(1) Boy                      1.00000000
## nodeofactor_layer.1.GENC.(1) Boy                      0.09731710
## nodeofactor_layer.2.GENC.(1) Boy                      0.57130275
```

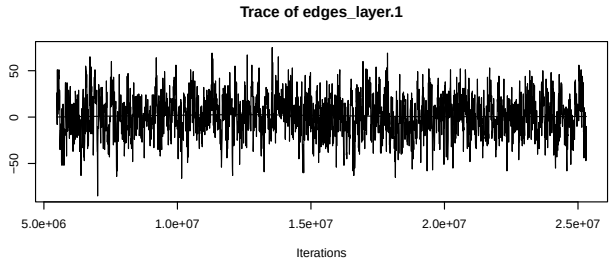
```
## mutual.same.layer.mem.1                                0.06026611
## mutual.same.layer.mem.2                                0.38317953
## duplexdyad.e                                           0.37934795
## duplexdyad.f                                           0.39248136
## duplexdyad.g                                           0.29877436
## duplexdyad.h                                           0.30419550
##                                 nodeofactor_layer.1.GENC.(1) Boy
## edges_layer.1                                          0.76499612
## edges_layer.2                                          0.10169272
## nodeifactor_layer.1.GENC.(1) Boy                       0.71984280
## nodeifactor_layer.2.GENC.(1) Boy                       0.09731710
## nodeofactor_layer.1.GENC.(1) Boy                       1.00000000
## nodeofactor_layer.2.GENC.(1) Boy                       0.11995170
## mutual.same.layer.mem.1                                0.55945384
## mutual.same.layer.mem.2                                0.05884009
## duplexdyad.e                                           0.18009888
## duplexdyad.f                                           0.16650001
## duplexdyad.g                                           0.15695372
## duplexdyad.h                                           0.06145955
##                                 nodeofactor_layer.2.GENC.(1) Boy
## edges_layer.1                                           0.1188729
## edges_layer.2                                           0.7618688
## nodeifactor_layer.1.GENC.(1) Boy                        0.1199181
## nodeifactor_layer.2.GENC.(1) Boy                        0.5713028
## nodeofactor_layer.1.GENC.(1) Boy                        0.1199517
## nodeofactor_layer.2.GENC.(1) Boy                        1.0000000
## mutual.same.layer.mem.1                                 0.0712282
## mutual.same.layer.mem.2                                 0.3225094
## duplexdyad.e                                            0.4198769
## duplexdyad.f                                            0.4017109
## duplexdyad.g                                            0.3242717
## duplexdyad.h                                            0.2587300
##                                 mutual.same.layer.mem.1
```

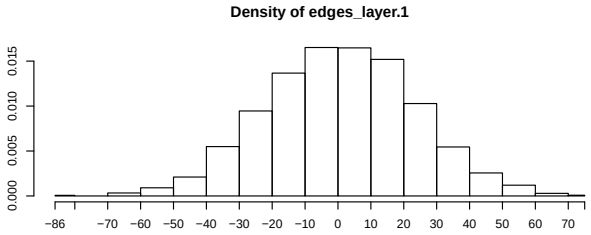
```
## edges_layer.1                                0.74497510
## edges_layer.2                                0.10622785
## nodeifactor_layer.1.GENC.(1) Boy             0.51790216
## nodeifactor_layer.2.GENC.(1) Boy             0.06026611
## nodeofactor_layer.1.GENC.(1) Boy             0.55945384
## nodeofactor_layer.2.GENC.(1) Boy             0.07122820
## mutual.same.layer.mem.1                      1.00000000
## mutual.same.layer.mem.2                      0.08307231
## duplexdyad.e                                 0.26018738
## duplexdyad.f                                 0.22819761
## duplexdyad.g                                 0.35088389
## duplexdyad.h                                 0.14022775
##                                 mutual.same.layer.mem.2 duplexdyad.e
## edges_layer.1                                0.11247989    0.2777288
## edges_layer.2                                0.53884653    0.5879764
## nodeifactor_layer.1.GENC.(1) Boy             0.06137669    0.1381044
## nodeifactor_layer.2.GENC.(1) Boy             0.38317953    0.3793479
## nodeofactor_layer.1.GENC.(1) Boy             0.05884009    0.1800989
## nodeofactor_layer.2.GENC.(1) Boy             0.32250944    0.4198769
## mutual.same.layer.mem.1                      0.08307231    0.2601874
## mutual.same.layer.mem.2                      1.00000000    0.5686243
## duplexdyad.e                                 0.56862432    1.0000000
## duplexdyad.f                                 0.57784133    0.7914543
## duplexdyad.g                                 0.47773197    0.8490986
## duplexdyad.h                                 0.82652130    0.6929612
##                                 duplexdyad.f duplexdyad.g duplexdyad.h
## edges_layer.1                      0.2672106    0.2594651   0.14625026
## edges_layer.2                      0.5815930    0.4607318   0.44553698
## nodeifactor_layer.1.GENC.(1) Boy   0.1807354    0.1626322   0.06713297
## nodeifactor_layer.2.GENC.(1) Boy   0.3924814    0.2987744   0.30419550
## nodeofactor_layer.1.GENC.(1) Boy   0.1665000    0.1569537   0.06145955
## nodeofactor_layer.2.GENC.(1) Boy   0.4017109    0.3242717   0.25873001
## mutual.same.layer.mem.1            0.2281976    0.3508839   0.14022775
```

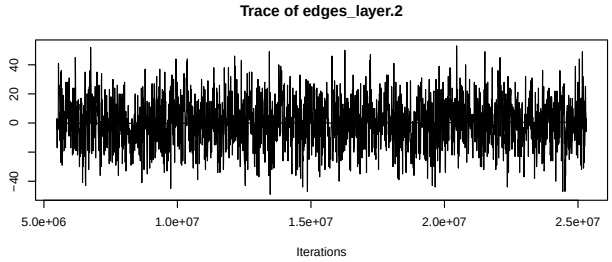
```
## mutual.same.layer.mem.2            0.5778413    0.4777320   0.82652130
## duplexdyad.e                       0.7914543    0.8490986   0.69296121
## duplexdyad.f                       1.0000000    0.8446017   0.68529601
## duplexdyad.g                       0.8446017    1.0000000   0.66833648
## duplexdyad.h                       0.6852960    0.6683365   1.00000000
##
## Sample statistics auto-correlation:
## Chain 1
##           edges_layer.1 edges_layer.2 nodeifactor_layer.1.GENC.(1) Boy
## Lag 0        1.00000000   1.000000000                        1.00000000
## Lag 8192     0.39380852   0.183817117                        0.38973695
## Lag 16384    0.29378797   0.086515205                        0.26821892
## Lag 24576    0.20119045   0.046791439                        0.19351317
## Lag 32768    0.13181348  -0.008239868                        0.14731383
## Lag 40960    0.07017867   0.038198666                        0.09450595
##           nodeifactor_layer.2.GENC.(1) Boy nodeofactor_layer.1.GENC.(1) Boy
## Lag 0                        1.000000000                        1.00000000
## Lag 8192                     0.116129077                        0.34776334
## Lag 16384                    0.047577302                        0.27143710
## Lag 24576                    0.054970597                        0.18531527
## Lag 32768                   -0.001218838                        0.11909352
## Lag 40960                    0.012171140                        0.04632327
##           nodeofactor_layer.2.GENC.(1) Boy mutual.same.layer.mem.1
## Lag 0                         1.00000000               1.0000000
## Lag 8192                      0.12331761               0.6577932
## Lag 16384                     0.04914506               0.4340840
## Lag 24576                     0.04146828               0.3028731
## Lag 32768                    -0.02226400               0.1798402
## Lag 40960                     0.04850819               0.1033392
##           mutual.same.layer.mem.2 duplexdyad.e duplexdyad.f duplexdyad.g
## Lag 0                1.000000000   1.00000000   1.00000000   1.00000000
## Lag 8192             0.334780414   0.38667236   0.37520561   0.43731663
## Lag 16384            0.158808764   0.22825699   0.21038798   0.24342718
```

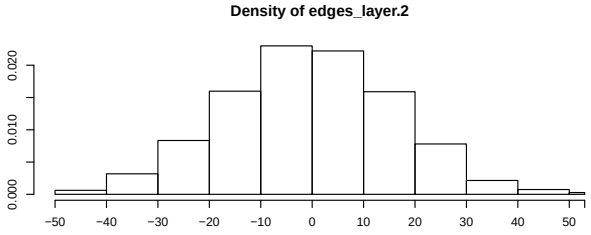
```
## Lag 24576             0.060227023   0.13494073   0.13699829   0.14608732
## Lag 32768             0.044995350   0.08240238   0.08792215   0.11418430
## Lag 40960             0.004322421   0.07093747   0.06463551   0.08575568
##           duplexdyad.h
## Lag 0       1.00000000
## Lag 8192    0.47538956
## Lag 16384   0.23611561
## Lag 24576   0.11622765
## Lag 32768   0.05736225
## Lag 40960   0.03260316
##
## Sample statistics burn-in diagnostic (Geweke):
## Chain 1
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##                    edges_layer.1                    edges_layer.2
##                          -0.1297                           0.7991
## nodeifactor_layer.1.GENC.(1) Boy nodeifactor_layer.2.GENC.(1) Boy
##                           1.1345                           0.7033
## nodeofactor_layer.1.GENC.(1) Boy nodeofactor_layer.2.GENC.(1) Boy
##                           0.7441                           0.7362
##          mutual.same.layer.mem.1          mutual.same.layer.mem.2
##                           0.6847                           0.7544
##                     duplexdyad.e                     duplexdyad.f
##                           0.4339                           1.0034
##                     duplexdyad.g                     duplexdyad.h
##                           0.9412                           0.9302
##
## Individual P-values (lower = worse):
##                    edges_layer.1                    edges_layer.2
##                        0.8967872                        0.4242039
```

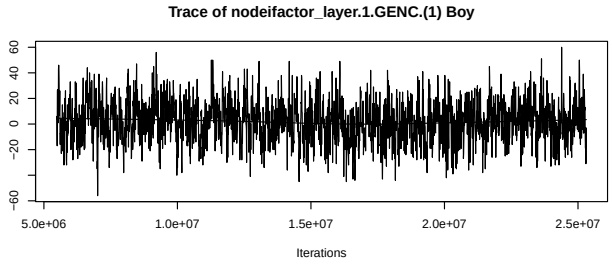
```
## nodeifactor_layer.1.GENC.(1) Boy nodeifactor_layer.2.GENC.(1) Boy
##                        0.2565849                        0.4818483
## nodeofactor_layer.1.GENC.(1) Boy nodeofactor_layer.2.GENC.(1) Boy
##                        0.4567995                        0.4615933
##          mutual.same.layer.mem.1          mutual.same.layer.mem.2
##                        0.4935088                        0.4506109
##                     duplexdyad.e                     duplexdyad.f
##                        0.6643883                        0.3156705
##                     duplexdyad.g                     duplexdyad.h
##                        0.3466122                        0.3522612
## Joint P-value (lower = worse):  0.2017476 .
```

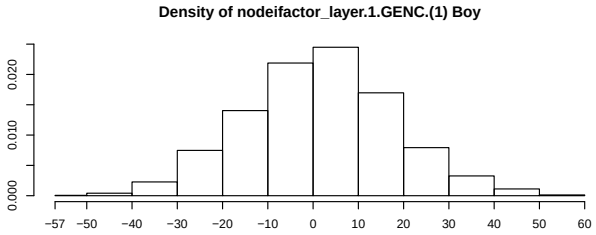


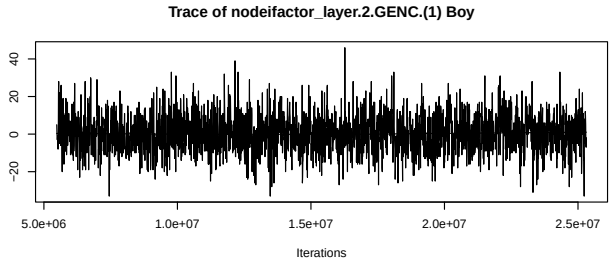


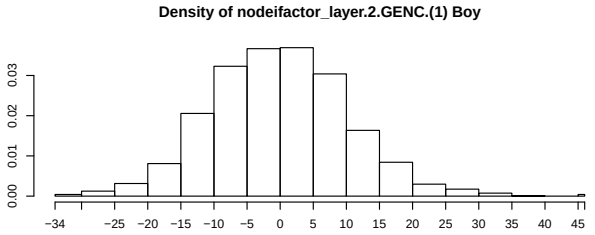


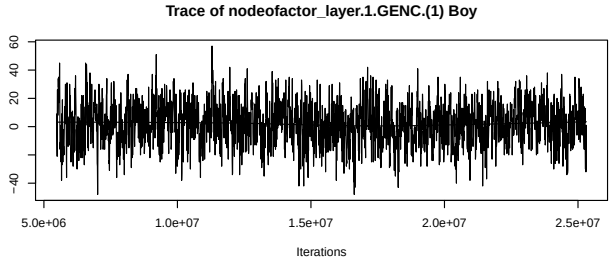


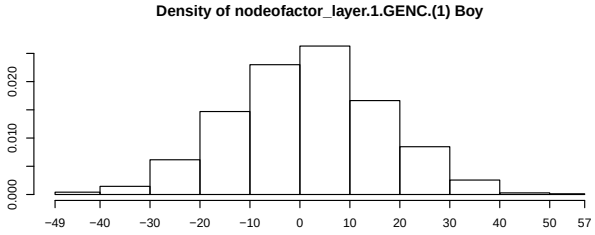


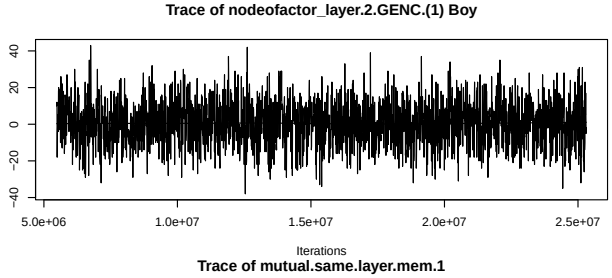


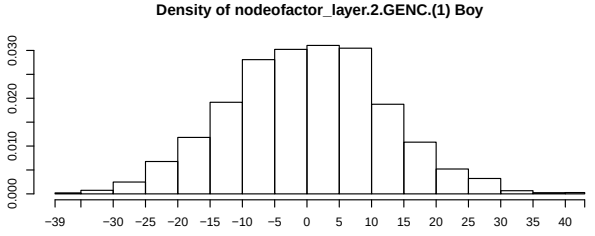

Trace of nodeifactor_layer.2.GENC.(1) Boy
Iterations

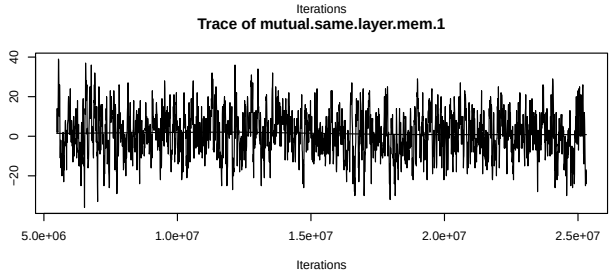

Density of nodeifactor_layer.2.GENC.(1) Boy

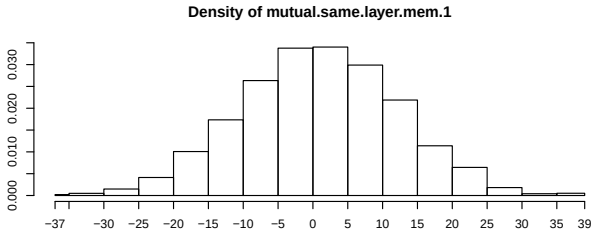

Trace of nodeofactor_layer.1.GENC.(1) Boy
Iterations

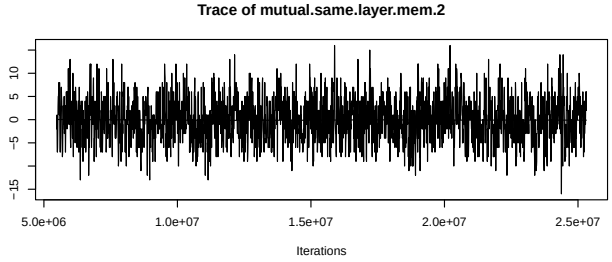

Density of nodeofactor_layer.1.GENC.(1) Boy

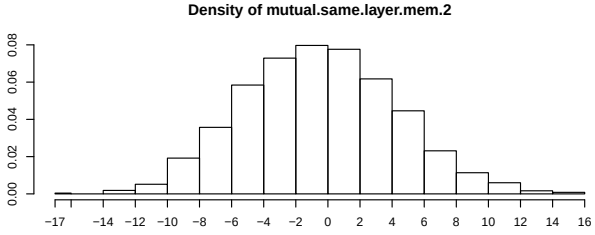

Trace of nodeofactor_layer.2.GENC.(1) Boy
Iterations

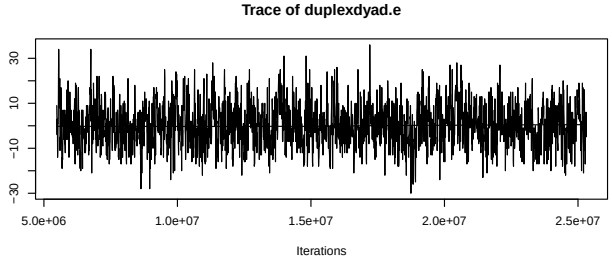

Density of nodeofactor_layer.2.GENC.(1) Boy

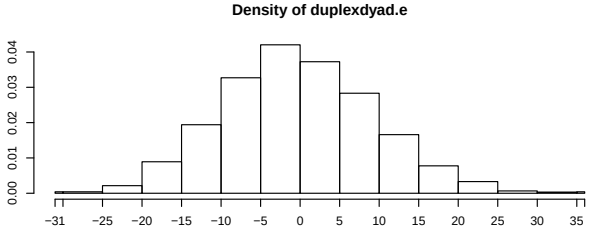

Trace of mutual.same.layer.mem.1
Iterations

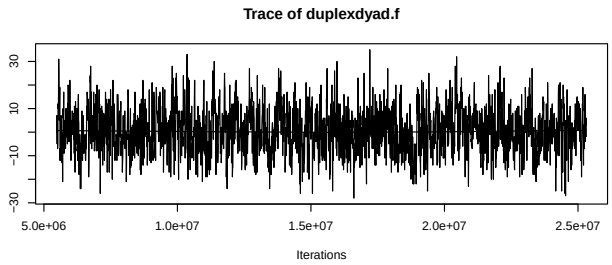

Density of mutual.same.layer.mem.1

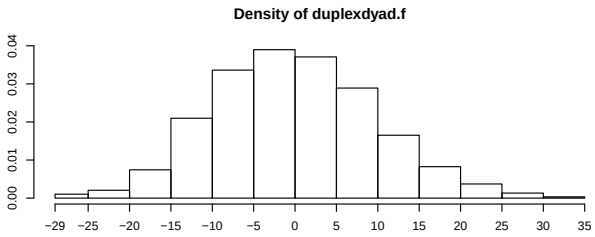

Trace of mutual.same.layer.mem.2
Iterations

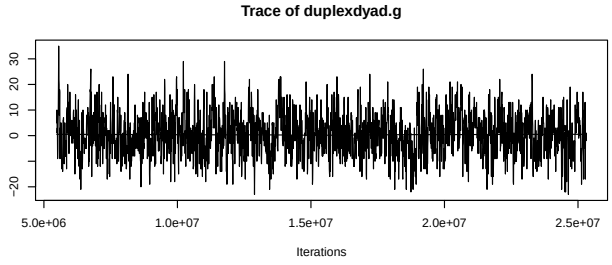

Density of mutual.same.layer.mem.2

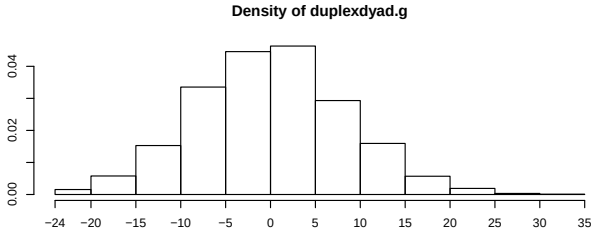

Trace of duplexdyad.e
Iterations

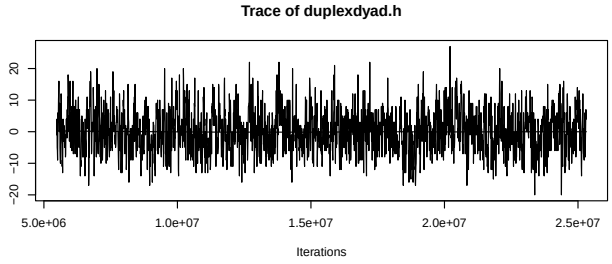

Density of duplexdyad.e

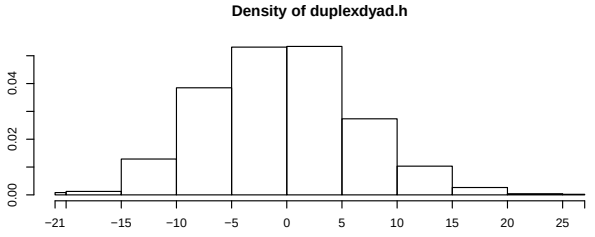




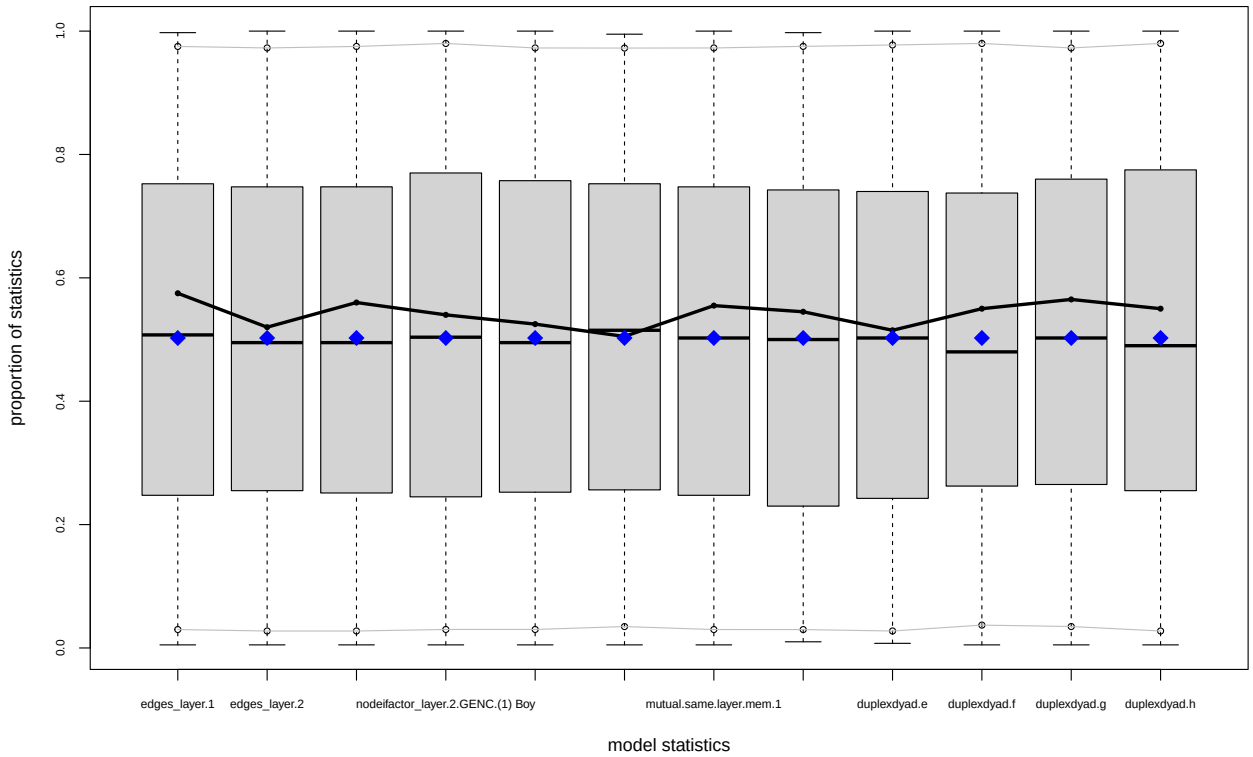


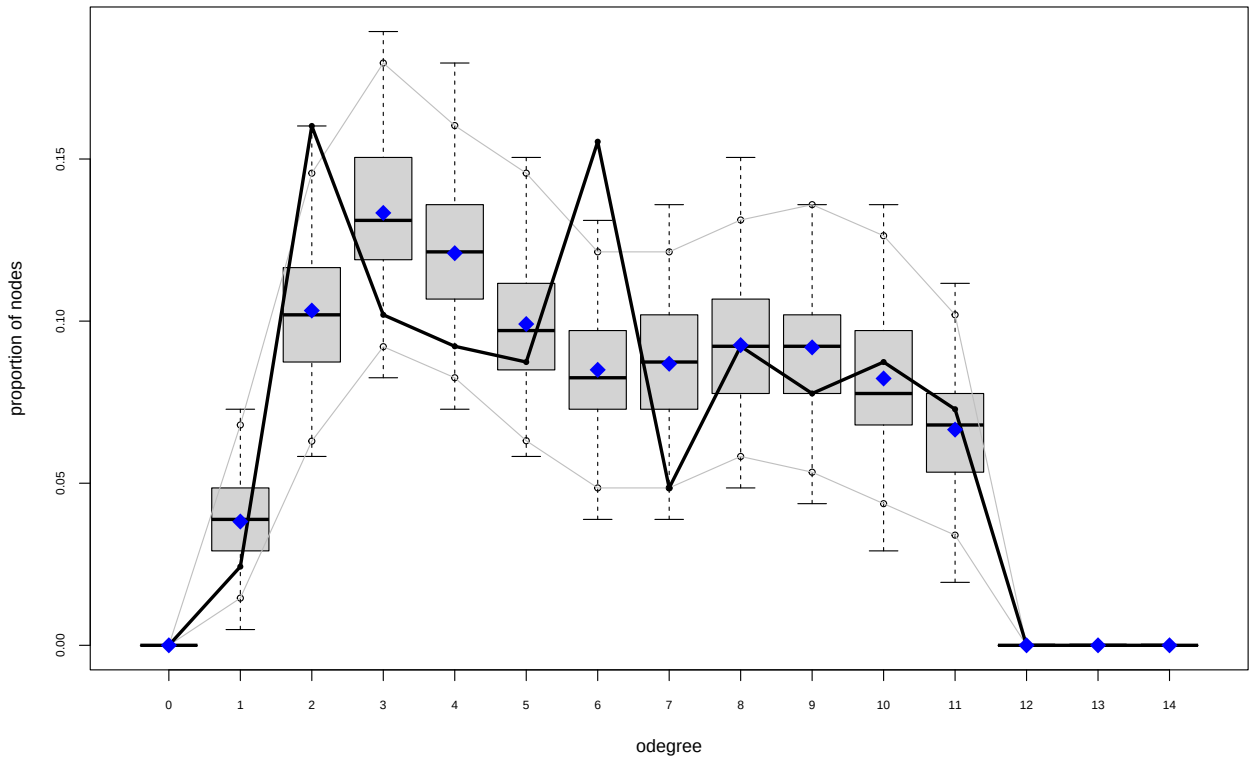


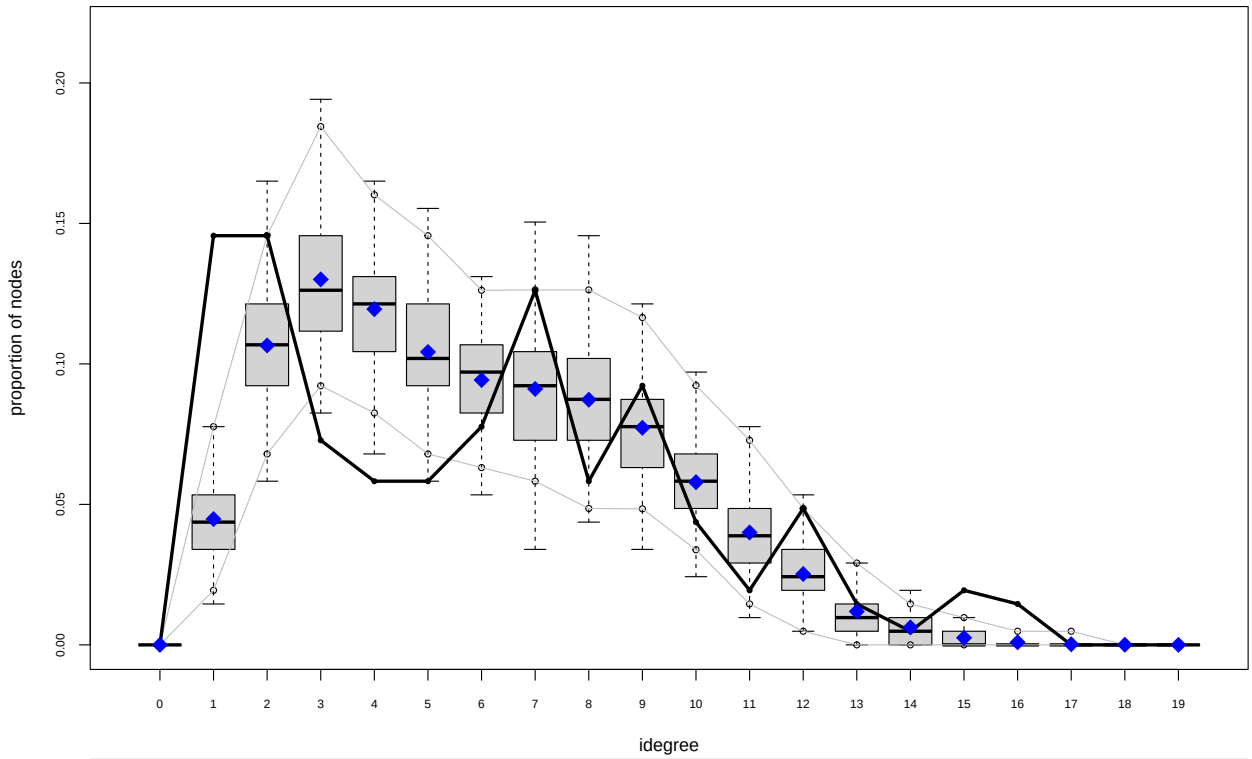


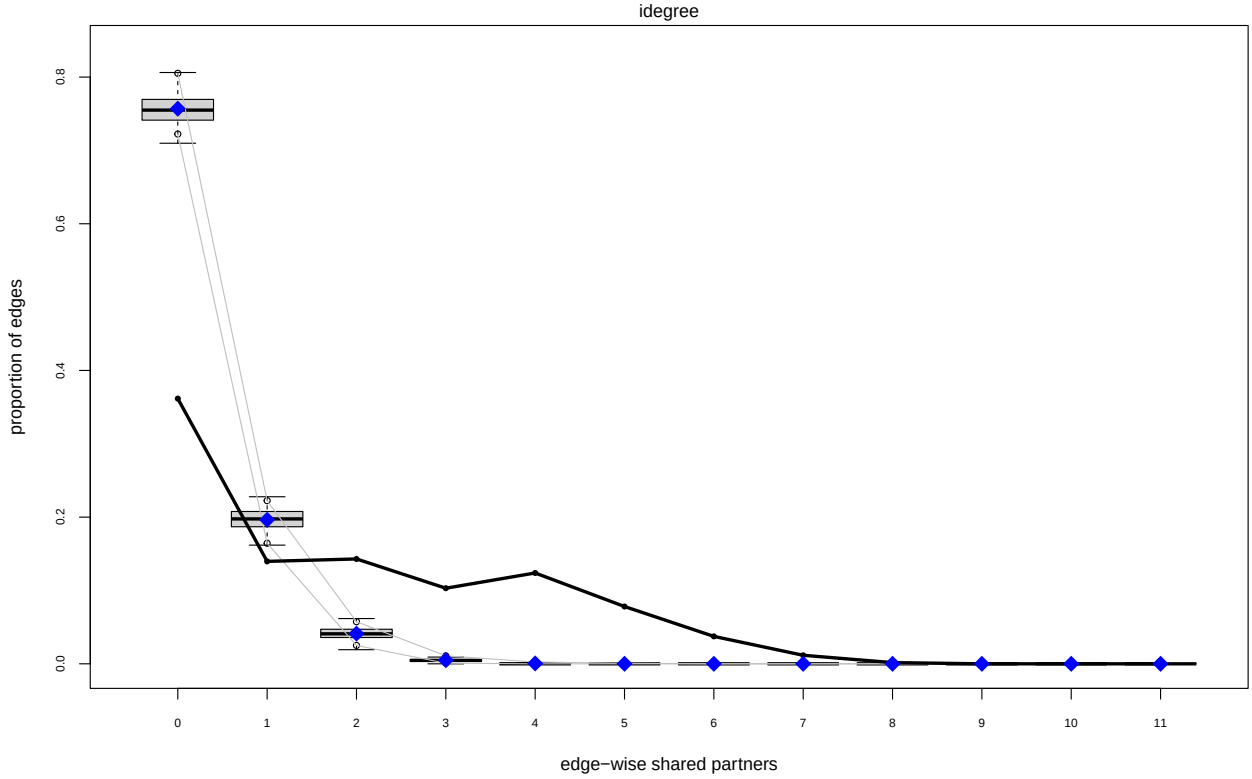


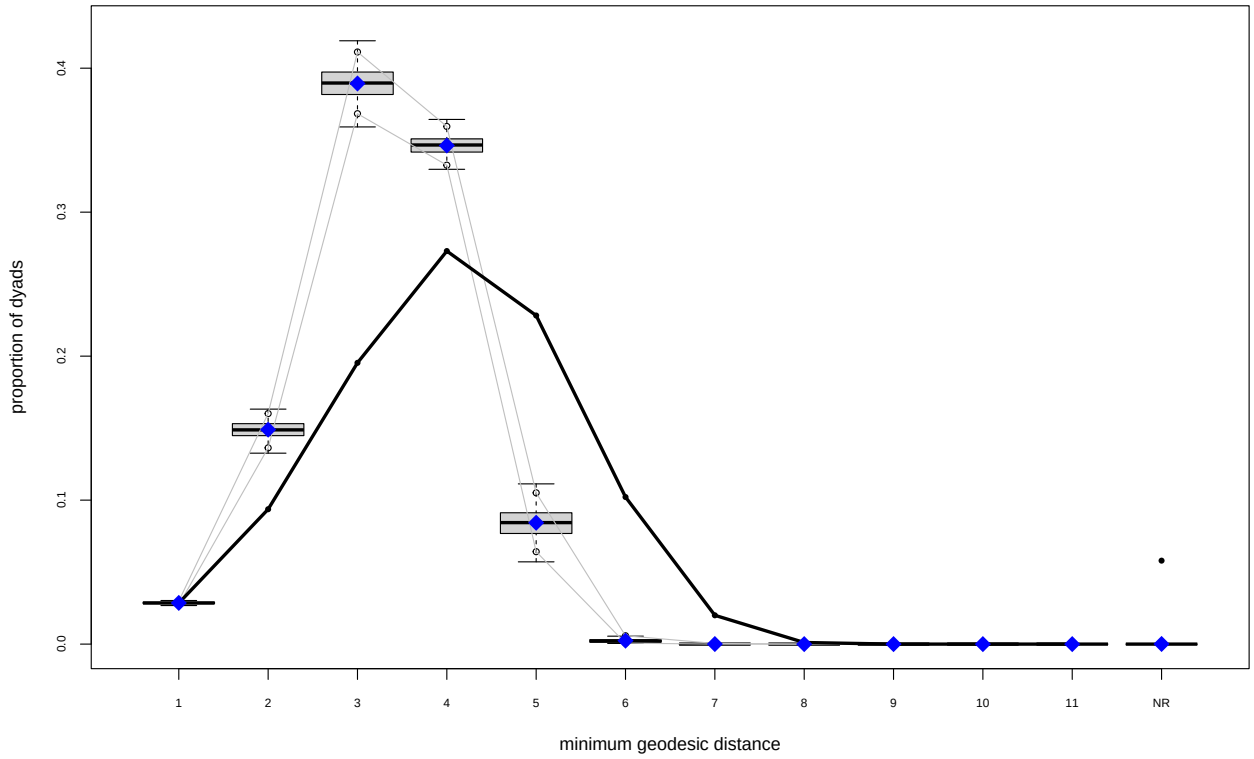

```
##
## MCMC diagnostics shown here are from the last round of simulation, prior to computation of final para
gof <- gof(mmodel.cross, verbose=T, burnin=1e+7, interval=1e+5, control = control.gof.ergm(nsim = 200))
plot(gof)
```


proportion of statistics
0.0
0.2
0.4
0.6
0.8
1.0
edges_layer.1
edges_layer.2
nodeifactor_layer.2.GENC.(1) Boy
mutual.same.layer.mem.1
duplexdyad.e
duplexdyad.f
duplexdyad.g
duplexdyad.h
model statistics


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


proportion of nodes
0.00
0.05
0.10
0.15
0.20
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
idegree


proportion of edges
0.0
0.2
0.4
0.6
0.8
0 1 2 3 4 5 6 7 8 9 10 11
edge−wise shared partners




```
(gof)
```

```
##
## Goodness-of-fit for in-degree
##
##            obs min   mean max MC p-value
## idegree1    30   3  9.225  18       0.00
## idegree2    30  12 21.955  34       0.07
## idegree3    15  17 26.795  45       0.00
## idegree4    12  14 24.615  37       0.00
## idegree5    12   8 21.475  36       0.02
## idegree6    16  11 19.420  32       0.45
## idegree7    26   7 18.765  31       0.12
## idegree8    12   9 17.975  30       0.20
## idegree9    19   7 15.915  27       0.42
## idegree10    9   3 11.930  21       0.40
## idegree11    4   0  8.240  17       0.13
```

```
## idegree12  10   1  5.195  11       0.07
## idegree13   3   0  2.450   6       0.86
## idegree14   1   0  1.280   4       1.00
## idegree15   4   0  0.520   3       0.00
## idegree16   3   0  0.190   2       0.00
## idegree17   0   0  0.045   1       1.00
## idegree18   0   0  0.010   1       1.00
##
## Goodness-of-fit for out-degree
##
##           obs min   mean max MC p-value
## odegree1    5   1  7.865  15       0.37
## odegree2   33  12 21.270  33       0.01
## odegree3   21  17 27.475  39       0.18
## odegree4   19  12 24.915  38       0.23
## odegree5   18   9 20.415  32       0.73
## odegree6   32   8 17.505  29       0.00
## odegree7   10   8 17.890  28       0.10
## odegree8   19  10 19.075  35       1.00
## odegree9   16   8 18.930  32       0.55
## odegree10  18   6 16.955  28       0.85
## odegree11  15   4 13.705  23       0.75
##
## Goodness-of-fit for edgewise shared partner
##
##      obs min    mean max MC p-value
## esp0 435 834 914.190 996       0.00
## esp1 168 162 237.385 282       0.01
## esp2 172  20  49.825  84       0.00
## esp3 124   0   6.010  38       0.00
## esp4 149   0   0.635   5       0.00
## esp5  94   0   0.055   3       0.00
## esp6  45   0   0.000   0       0.00
```

```
## esp7  14   0   0.000   0       0.00
## esp8   2   0   0.000   0       0.00
##
## Goodness-of-fit for minimum geodesic distance
##
##       obs   min      mean   max MC p-value
## 1    1203  1131  1208.100  1305       0.93
## 2    3956  5600  6290.820  7093       0.00
## 3    8250 15169 16442.410 18002       0.00
## 4   11529 13578 14622.500 15389       0.00
## 5    9638  2230  3558.980  4800       0.00
## 6    4314    20   101.295   285       0.00
## 7     845     0     1.780    74       0.00
## 8      47     0     0.035     4       0.00
## Inf  2448     0     4.080   408       0.00
##
## Goodness-of-fit for model statistics
##
##                                    obs min    mean max MC p-value
## edges_layer.1                      738 684 742.530 793       0.88
## edges_layer.2                      259 211 259.570 306       1.00
## nodeifactor_layer.1.GENC.(1) Boy 374 331 376.070 422       0.92
## nodeifactor_layer.2.GENC.(1) Boy 106  83 106.515 136       1.00
## nodeofactor_layer.1.GENC.(1) Boy 388 338 388.505 427       1.00
## nodeofactor_layer.2.GENC.(1) Boy 146 110 145.765 172       1.00
## mutual.same.layer.mem.1          213 188 214.820 242       0.98
## mutual.same.layer.mem.2           23  12  22.950  35       1.00
## duplexdyad.e                      80  60  80.515 108       1.00
## duplexdyad.f                      82  61  82.115 110       1.00
## duplexdyad.g                      54  35  54.135  76       0.94
## duplexdyad.h                      27  11  26.730  44       1.00
```