Universität Konstanz
Fachbereich Informatik und Informationswissenschaften
Projektbericht zu Netzwerkmodellen
Prof. Dr. Ulrik Brandes
WS 2009/2010
Betreuer: Dr. Jürgen Lerner

Analysing A & I and the Mirror Network with ERGMs

Zusammenfassung

Attract & Introduce wurde von Fowler und anderen in [1] eingeführt und mit Hilfe von Add Health gesammelten Daten mit anderen Modellen verglichen. Der Vergleich mit dem Exponential Random Graph Model (ERGM) wird hier kritisch hinterfragt und es wird mit Hilfe von zahlreichen Tests eine Aussage darüber getroffen wieviel schlechter die ERGMs im Vergleich zu Attract & Introduce überhaupt sein können. Anschließend wird die Spiegel Netzwerk Methode etwas genauer untersucht und versucht eine bessere Parametrisierung für die ERGMs zu finden.

von
Arlind Nocaj
arlind.nocaj{at}uni-konstanz.de
07.04.2010

Inhaltsverzeichnis

1 Einführung

Die ERGM-Klasse bietet sehr vielseitige Modelle mit denen man verschiedenste Netzwerke simulieren kann. Ein solches Modell bezüglich des Parametervektors η ist definiert als:

$$P_\eta(Y = y) = \frac{exp\{\eta^t g(y)\}}{\kappa(\eta)} \tag{1}$$

wobei $\kappa(\eta) = \sum_z exp\{\eta^t g(z)\}, z \in \Omega$ gilt.
Mehr Informationen hierzu können unter [6] nachgelesen werden. In diesem Bericht werden zudem weitere Begriffe von Methoden wie z.B. A & I und Spiegel Netzwerk benutzt, die als bekannt vorausgesetzt werden. Die notwendigen Grundlagen können unter [4, 1, 3] nachgelesen werden.

Entscheidend bei der Anwendung sind folgende Punkte:

- Statistiken
- Parameter
- Konvergenz beim Ziehen zufälliger Graphen

Statistiken: Statistiken beschreiben bestimmte Eigenschaften und Strukturen eines Netzwerkes und definieren durch ihre Gewichtung die Wahrscheinlichkeitsverteilung. Sie spielen damit eine sehr wichtige Rolle bei den ERGMs. Je nachdem welche Statistiken benutzt werden, entsteht ein gutes oder degeneriertes Modell. Dabei gibt es einige Statistiken die früher benutzt wurden, von denen klar ist, dass sie sehr leicht zur Degeneration des Modells führen. Snijders und andere beschreiben dieses Problem in [6] ausführlich.
Eine dieser Statistiken ist der *triadcensus*.

Fowler benutzt folgende Statistiken für die ERGMs:

- nodeicov("attract")
- triadcensus

nodeicov("attract"): Zählt die Eingangskanten und gewichtet diese mit dem Attributwert, in diesem Fall der Attract-Faktor, des jeweiligen Knotens auf den die Kante zeigt.

triadcensus: Der *triadcensus* beschreibt alle verschiedenen Dreiecksstrukturen die in einem Netzwerk auftreten können. In Abb. 1 sind die verschiedenen Strukturen zu sehen wobei die rot markierten von Fowler benutzt wurden.

Um das Degenerieren der Modelle zu verhindern, werden in den von mir durchgeführten Tests zudem noch folgende Statistiken benutzt die von Hunter, Goodreau und Handcock [3] empfohlen werden:

- gwidegree (geomatrically weighted indegree)
- gwoutdegree (geomatrically weighted outdegree)
- gwdsp (geomatrically weighted dyad-wise shared partners)

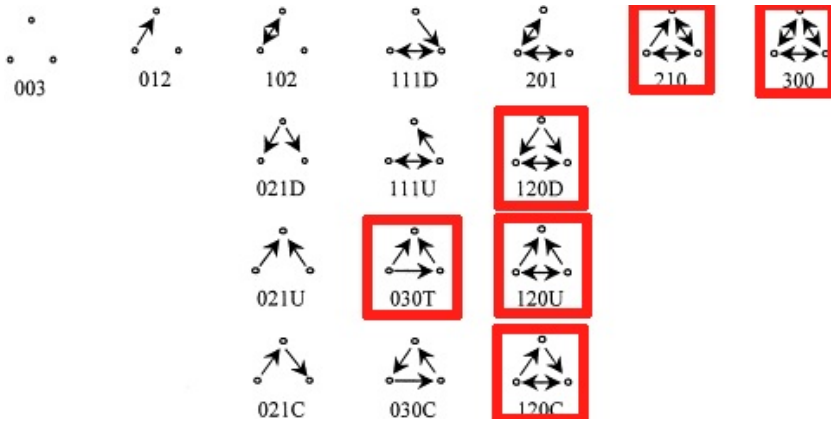
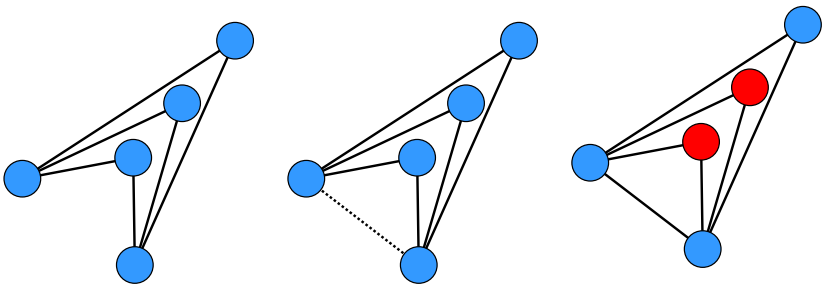


Abbildung 1: verschiedenen Strukturen von triadcensus

- gwesp (geomatrically weighted edge-wise shared partners)

Diese Statistiken sind in dem ergm-package für R [2] bereits enthalten und sind in der Regel viel besser für das Modellieren von Netzwerken geeignet als die früheren Statistiken, welche keine geometrische Gewichtung haben. Eine genauere Beschreibung der Statistiken findet sich in der Dokumentation des R-Packages [2].

itriangle Statistik: Um die Introduce-Eigenschaft als solche in einer Statistik zu berücksichtigen und damit zumindest teilweise zu modellieren wurde von mir eine zusätzliche Statistik entwickelt und implementiert. *itriangle* ist eine Statistik, die für eine Kante die gemeinsamen Nachbarn ihrer Endknoten zählt und dabei nur die berücksichtigt die als Knotenattribut für *introduce* den Wert 1 haben.



(a) dyad-wise shared partners

(b)edge-wise shared partners

(c) itriangle, rote Knoten haben die Introduce-Eigenschaft

2 Allgemein

Es gibt einige Punkte in *Model of genetic variation in human social networks* [1] die problematisch sind, bzw. von denen nicht klar ist, ob sie sorgfältig bearbeitet wurden um einem fairen Vergleich zwischen Attract & Introduce und den ERGMs zu bilden:

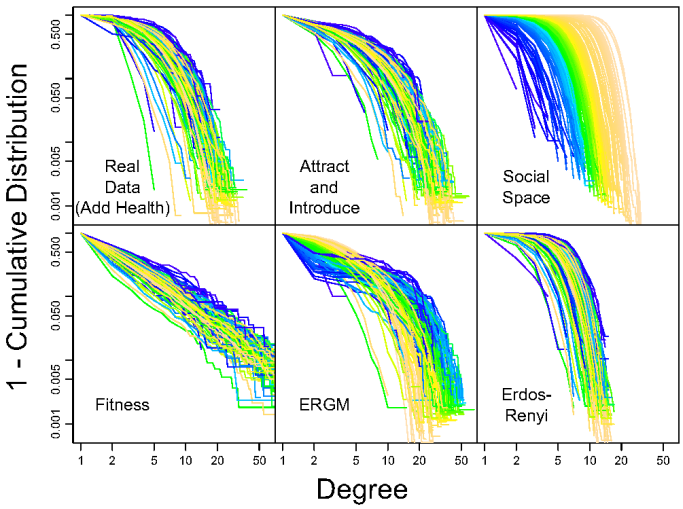


Fig. 2. Comparison of degree distributions in real social networks (within each of the 146 schools in the Add Health sample) and like-sized simulated networks based on 5 models, the Attract and Introduce model; a social space model (8); a fitness model (7); an ERGM (12); and an Erdos–Renyi random network (27). In *Upper Left*, each line indicates the in-degree distribution for each school in Add Health. In other images, line indicates the degree distribution in one simulation that assumes the same number of nodes and edges as each of the 146 schools. The color of each line indicates the size of the network (number of nodes) with yellow shades for small, green for medium, and blue for large networks (total range 9 to 2,724, mean 752). The fitness

Abbildung 2: Degree-Vergleich aus [1]

1. Indegree wird mit Degree verglichen
2. schlechte Statistiken zum Nachbilden von Netzwerken gewählt
3. ERGM-Parameter wurden nicht geschützt
4. Burnin

In Abb. 2 wird aus der Beschreibung klar, dass für die echten Daten der Eingangsgrad benutzt wurde, während für die Modelle lediglich der Grad erwähnt wird. Eventuell wurde der Knotengrad nur bei den Modellen genommen, bei denen keine gerichteten Netzwerke entstehen, aber dann wäre es sinnvoll auch den Knotengrad der realen Netzwerke für diese Modelle zu benutzen und nicht den Eingangsgrad.

Ein weiteres Problem besteht bei der Verwendung der Statistiken, welches in den nachfolgenden Teil beschrieben wird.

3 Spezieller Test

Da die Daten der echten Netzwerke nicht vorhanden waren, konnte der Vergleich von Fowler nicht rekonstruiert werden. Das gute Abschneiden von Attract &

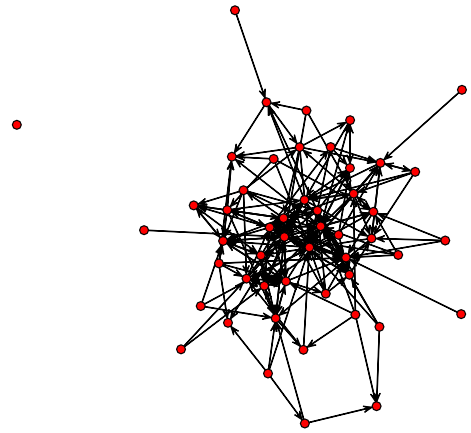


Abbildung 3: spezielles Netzwerk mit alpha=0.9, beta=0.3, 50 Knoten, 210 Kanten

Introduce wird genutzt um mit einem ERGM welches A & I möglichst gut abbildet vielleicht bessere Ergebnisse zu erzielen.

Um ein besseres Verständnis für verschiedene ERGMs und daraus resultierende Netzwerke zu erhalten, wurde ein konkreter Graph mit der A & I - Methode generiert, der Quellcode für den A & I Generator findet sich im Anhang A.1.

Dieses spezielle Netzwerk, siehe Abb. 3, wurde anschließend etwas näher untersucht bzw. als ERGM modelliert.

Folgende Kombinationen von Statistiken wurden getestet um ein möglichst gutes ERGM zu erhalten:

1. edges + nodeicov
2. edges + nodeicov + triadcensus (Fowler)
3. edges + nodeicov + itriangle
4. edges + gwesp + gwdsp + gwidegree + gwodegree
5. edges + gwesp + gwdsp + gwidegree + gwodegree + nodeicov
6. edges + gwesp + gwdsp + nodeicov
7. edges + gwesp + gwdsp + nodeicov + itriangle

Abb. 4 zeigt die generierten Graphen der entsprechenden Modelle. Sehr auffällig ist hierbei, dass die Modelle aus der Statistikkombination (2), welches Fowler benutzt hat, eindeutig degenerieren. Ein einzelnes Degenerieren dieser Statistiken ist dabei auszuschließen, da mehrmals gefittet wurde und jedes mal das gleiche Ergebnis herauskam. Wenn die originalen Netzwerke aus den Add Health Daten *ähnlich* den A & I Netzwerken sind, so kann man definitiv sagen, dass die Statistiken aus (2) definitiv nicht sinnvoll sind, weil sie vollkommen degenerierte Modelle liefern. Dies ist nicht verwunderlich, Snijders und andere beschreiben dieses Problem der *alten Statistiken* sehr detailiert in [5, 6].

Bei der Modellierung ist insbesondere zu beachten, dass nur Graphen einer bestimmten Größe erwünscht sind. Um dies zu gewährleisten gibt es zwei Möglichkeiten. Die erste Möglichkeit ist das Verwenden der *edges* Statistik, welche die Wahrscheinlichkeitsverteilung dadurch direkt beeinflusst. Die zweite Möglichkeit besteht darin, beim Ziehen eines Netzwerkes aus dem Wahrscheinlichkeitsraum, *edges* als Constraint zu setzen wodurch die Markovkette beeinflusst wird. Es ist dabei zu beachten, dass *edges* als Constraint nicht immer die gewünschte Kantenanzahl liefert. Dies liegt wahrscheinlich daran, dass durch nicht Benutzen der edges Statistik einfach andere Graphen viel wahrscheinlicher sind. Wenn die Kantenanzahl mit Hilfe der Constraints einfach fixiert werden würde, würde der Wahrscheinlichkeitsraum dadurch ein völlig anderer sein.

Aus diesem Grund wurde *edges* als Statistik mit aufgenommen.

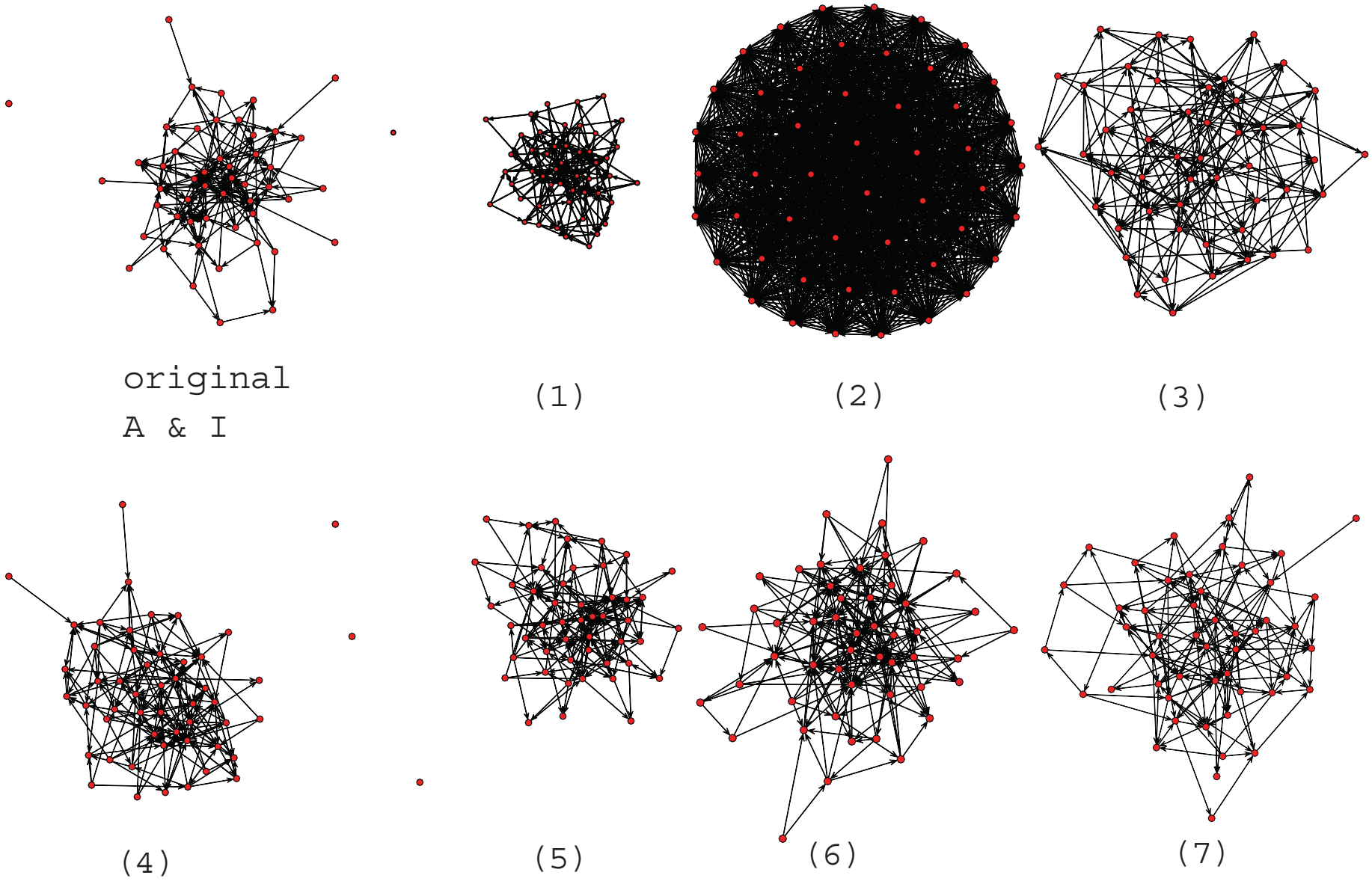


Abbildung 4: Generierte Graphen aus den jeweiligen ERGMs

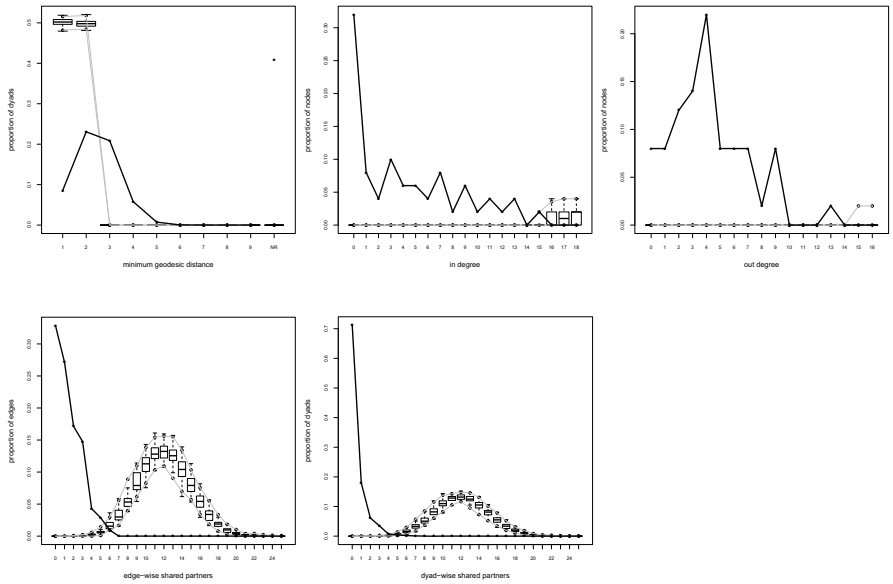


Abbildung 5: Statistikkombination (2), die Fowler benutzt hat

Einzelne generierte Netzwerke sagen natürlich nicht unbedingt etwas über das Modell an sich aus und es ist auch schwer optisch zu bestimmen wie gut nun ein Netzwerk einem anderen ähnelt. Um eine bessere Aussage über die Qualität der Modelle treffen zu können wurde daher die Goodness of Fit Methode von Hunter, Goodreau und Handcock [3] benutzt. Diese besteht darin die Verteilung verschiedner Statistiken, wie z.B. die minimale Pfaddistanz zweier Knoten oder Knotengrad, für das originale sowie für viele generierte Netzwerke des Modells in einem Schaubild darzustellen um sie leicht vergleichen zu können. Abb. 5 bestätigt ebenfalls das Degenerieren der Statistikkombination (2). Zur Überprüfung der Ähnlichkeit wurde die minimale Pfaddistanz zweier Knoten, der Eingangsgrad, der Ausgangsgrad, die Edge-wise shared partners und die dyd-wise shared partners als Statistik herangezogen. Diese Statistiken wurden gewählt, weil sie im Vergleich zu anderen Statistiken besonders viel Information über die Struktur des Graphen enthalten [3]. Natürlich kann die Ähnlichkeit gemessen an anderen Statistiken auch anders ausfallen.

Wenn man die Resultate der verschiedenen Statistiken betrachtet, siehe Anlage B, wird klar das die Statistikkombination (6) besonders gut geeignet ist um das konkrete Attract & Introduce Netzwerk zu simulieren. In Abb. 6 ist zu sehen wie für jede der Ähnlichkeitsstatistiken das originale Netzwerk relativ gut durch die Simulierten getroffen wird.

Fowler hat jedoch die Statistikkombination (2) verwendet, weswegen sich die Frage stellt ob sein exponetielles Modell ebenfalls degeneriert. Fowler beschreibt nicht wie er die Parameter gewählt hat. Da er für jeden triadcensus-Typ den Wert 2.7 benutzt hat und bei keiner einzigen Parameterschätzung einheitliche Werte ermittelt werden konnten, ist anzunehmen dass der Parameter 2.7 durch manuelles Testen ermittelt wurde.

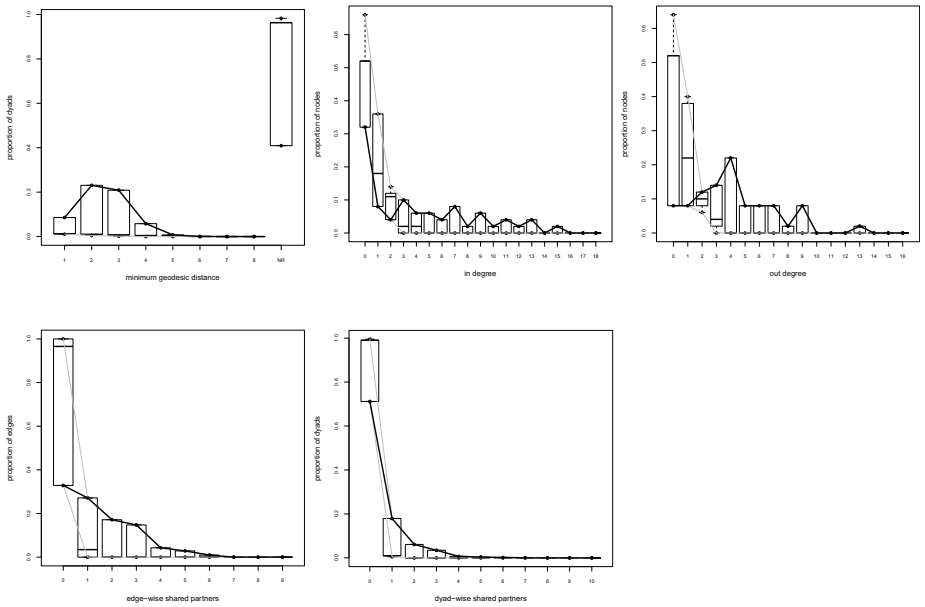


Abbildung 6: Statistikkombination (6) simuliert das A & I Netzwerk viel besser

Abb. 7 zeigt ein simuliertes Netzwerk mit den Parametern von Fowler. Wie zu erkennen ist degeneriert es mit seinen Parametern nur teilweise und nicht komplett. Insofern ist es besser, dass Fowler die Parameter manuell ermittelt hat, denn wenn er diese gefittet hätte, würde das Modell höchstwahrscheinlich komplett degenerieren.

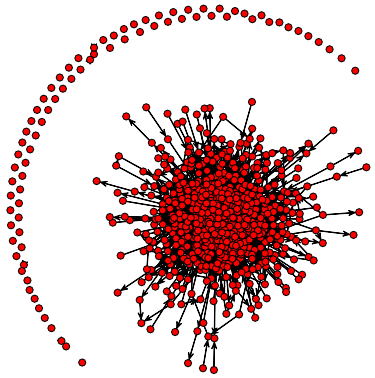


Abbildung 7: ERGM-Simulation mit Fowlers Parametern, degeneriert teilweise.

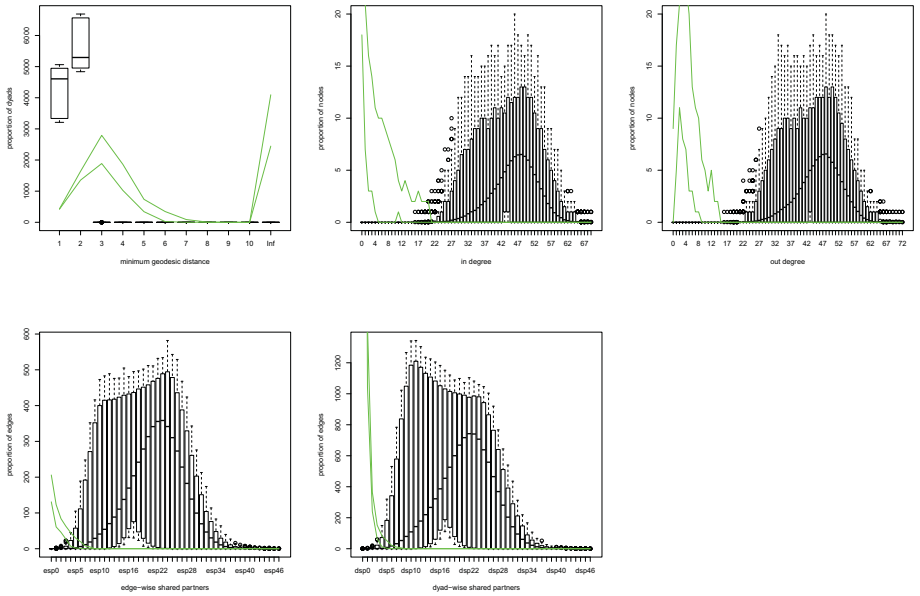


Abbildung 8: Statistikkombination (2) simuliert A & I sehr schlecht. Im grünen bereich liegen die A & I Netzwerke und die Boxplots stellen den Bereich dar in dem sich die Simulierten Netzwerke befinden.

4 Genereller Test

Auch wenn es sehr stark danach aussieht, dass die Statistikkombination (6) A & I Netzwerke sehr gut simuliert, so ist dies nur ein konkretes Beispiel, welches nicht benutzt werden sollte um eine generelle Aussage über die Qualität der Statistiken zu treffen.

Aus diesem Grund wurde die Goodness of Fit Methode von mir erweitert um statistisch eine Aussage über eine ganze Netzwerkklasse treffen zu können. Dabei wird nicht wie bei dem speziellen Test ein A & I Netzwerk zufälligt erzeugt und gefittet, sondern es werden sehr viele generiert und anschließend gefittet. Aus den daraus gewonnenen Modellen werden anschließend Netzwerke gezogen und mit den originalen verglichen indem der Bereich der Statistiken in einem Schaubild dargestellt wird.

Abb. 8 zeigt, dass die Statistikkombination (2) von Fowler A & I nicht gut approximieren. Die dazugehörigen Modelle degenerieren und erzeugen teilweise vollständige Graphen. Die neueren Statistiken hingegen liefern wie bei dem speziellen Test sehr gute Ergebnisse, was in Abb. 9 zu sehen ist. Alle verwendeten Vergleichsstatistiken werden sehr gut durch die Simulierten Netzwerke getroffen. Die Statistikkombination (6) ist somit durchaus gut geeignet um A & I Netzwerke zu simulieren. Die itriangle Statistik, siehe Anlage 8 , verändert die Darstellung nur minimal, sodass keine generelle Aussage getroffen werden kann. Hier könnte vielleicht eine geometrische Gewichtung der itriangle Statistik, wie bei den neueren Statistiken, bessere Ergebnisse bewirken.

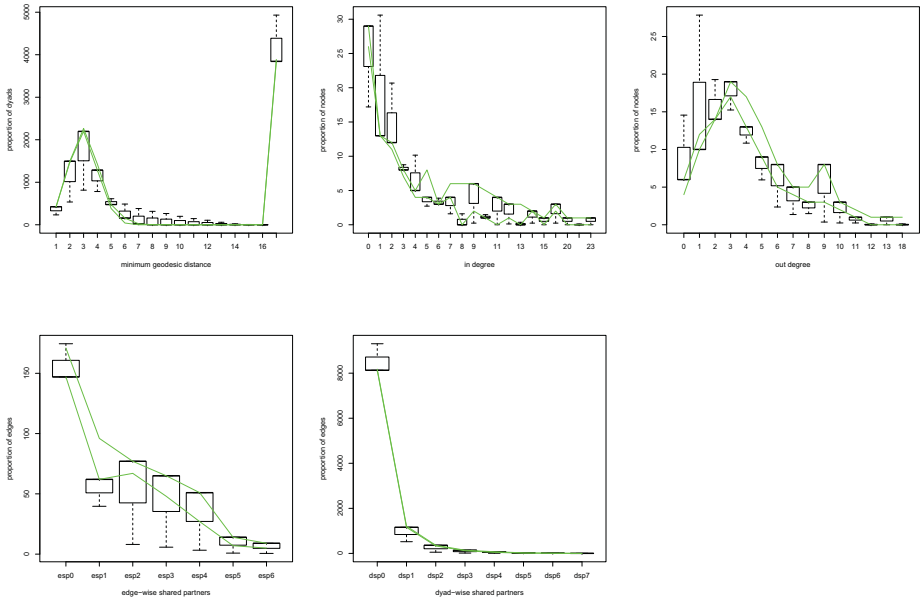


Abbildung 9: Statistikkombination (6) simuliert A & I sehr gut.

5 Spiegel Netzwerk

Fowler benutzt das Spiegel Netzwerk um damit zu messen wie gut bei einem Netzwerkmodell bestimmte Merkmale durch z.B. Knotenattribute bestimmt werden. Dies soll bestimmte Gene eines Menschen widerspiegeln die durch eine Studie von eineigen Zwillingen ermittelt wurden.

Zunächst stellt sich die Frage wie legitim diese Methode ist. Diese Methode versucht zwar die Vererbung von bestimmten Strukturen wie Indegree und Transitivität zu messen, jedoch besteht nicht unbedingt ein Bezug zu realen Netzwerken. Wenn z.B. ein Modell in der Spiegel Netzwerk - Methode sehr gute Ergebnisse liefert, so heisst dies noch lange nicht, dass die daraus entstehenden Netzwerke realistisch sind.

Nichtsdestotrotz wurde zunächst einmal überprüft ob durch das Verändern der Parameter des exponentiellen Modells eine höhere Vererbung im Spiegel Netzwerk erreicht werden kann.

Bestimmung der Parameter: Um die Parameter für die Statistikkombination (2), welche auch Fowler benutzt hat, zu bestimmen wurden viele A & I Netzwerke generiert und diese dann anschließend gefittet. Abb. 10 zeigt eine einzelne Statistik, wobei jeder Punkt eine Parameterschätzung für ein A & I Netzwerk ist. Es wurde jeweils der Mittelwert der Parameter, sowie der Parameter selbst probiert um damit höhere Vererbung in Fowlers Spiegel Netzwerk zu erhalten.

Die daraus resultierenden Vererbungen bzw. Korrelationen waren jedoch immer viel schlechter als unter Benutzung von Fowlers Parameter.

Die Netzwerkgröße wurde ebenfalls beachtet, indem Netzwerke unterschiedlicher Größe gefittet wurden um die optimalen Parameter zu wählen. Abb. 11

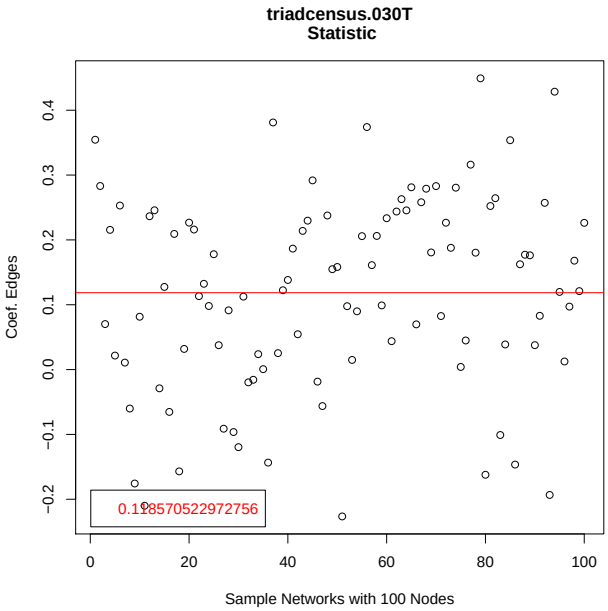


Abbildung 10: Parameter für bestimmten triadcensus-Typ liegt immer in einem bestimmten Bereich.

zeigt die lineare Abhängigkeit der jeweiligen Parameter zur Netzwerkgröße.

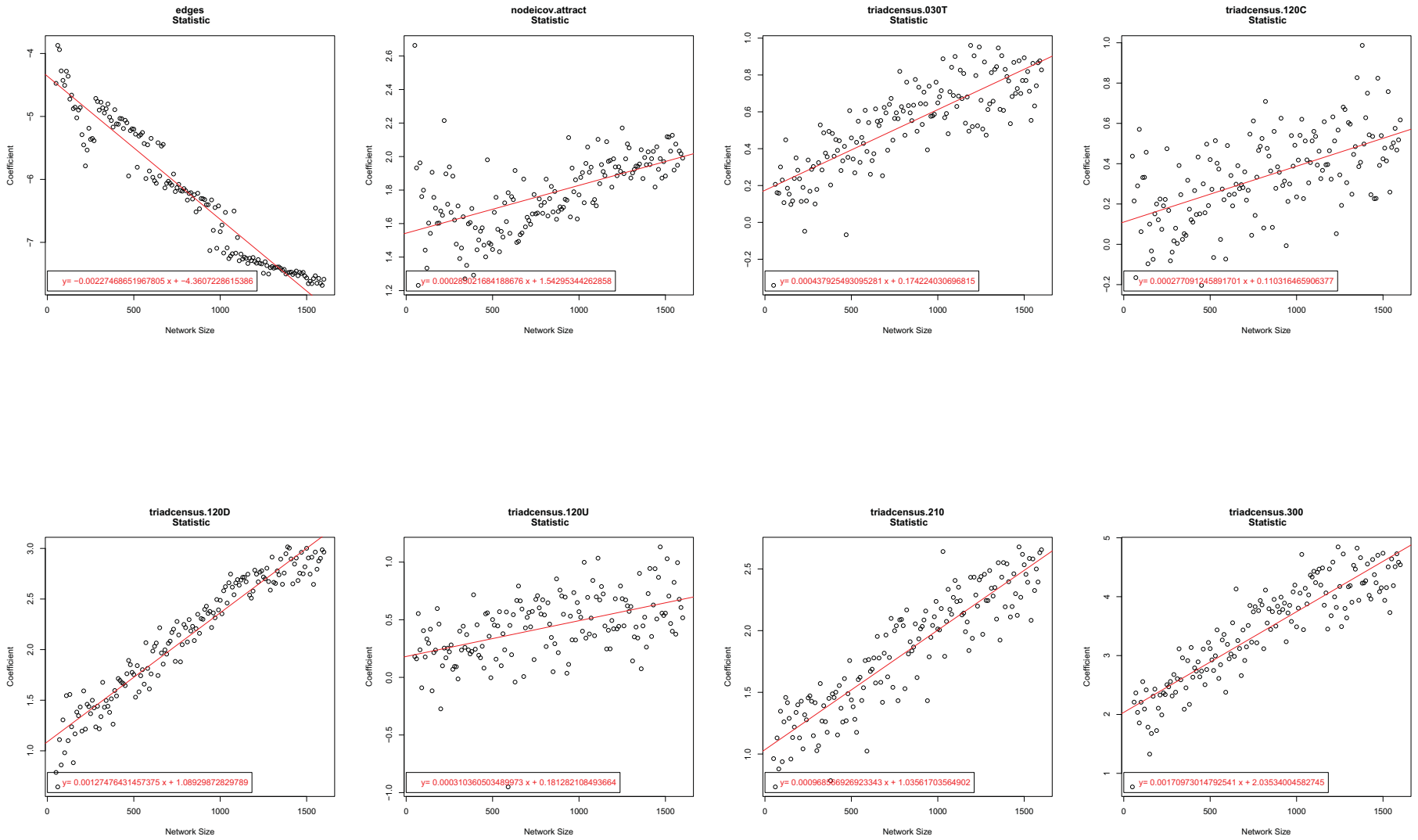


Abbildung 11: Ermittelte Parameter über die Netzwerkgröße (50 - 1600 Knoten)

An folgenden Punkten wurden sehr viele unterschiedliche Variationen ausprobier um eine Verbesserung der Korrelation beim Spiegel Netzwerk zu erreichen:

- verschiedene Netzwerkgrößen
- verschiedene Statistiken
- verschiedene Parameter (einzelne Schätzung, Mittelwert aus mehreren Schätzungen)

Trotz dieser Variationen war die Korrelation immer viel schlechter als die mit Fowlers Parameter. Die durch Schätzung ermittelten Parameter sind somit nicht für eine hohe Vererbung in der Spiegel Netzwerk Methode geeignet. Dies verwundert etwas, da durch die Statistikkombination (6) jedes A & I Netzwerk sehr gut simuliert werden konnte. Die Frage die sich nun stellt ist, was die Spiegel Netzwerk Methode für eine Aussage hat, wenn sie gute Ergebnisse liefert für Modelle die Netzwerke generieren die in der Realität nicht vorkommen, aber schlechte Ergebnisse für jene Modelle, die realitätsnahe Netzwerke erzeugen.

6 Zusammenfassung

ERGMs sind ein sehr mächtige Methode um Netzwerke zu modellieren. Dieses Potenzial kann jedoch lediglich genutzt werden, wenn man die Klasse genauer versteht und entsprechend sorgfältig auch die Parametrisierung durchführt. Ob Fowler für das Modellieren der realen Netzwerke nicht doch eine Parameterschätzung durchgeführt hat, kann nicht ausgeschlossen werden. Es erscheint jedoch sehr unwahrscheinlich dass er dies gemacht hat, wenn man betrachtet wie der gleiche Parameter für jeden triadcensus-Typ gewählt wurde. Was das Spiegel Netzwerk angeht, so kann man ihm nicht den Vorwurf machen, dass er schlechte Parameter gewählt hat. Wenn er jedoch die gleichen Parameter für das Simulieren der realen Netzwerke gewählt hat, so liefern diese keine realistischen Netzwerke. Wie die Tests gezeigt haben kann durch Verwendung von gwesp, gwdsp und nodeicov ein Attract & Introduce Netzwerk sehr gut simuliert werden. Es ist daher sehr wahrscheinlich, dass die originalen Netzwerke mit diesen Statistiken ebenfalls viel besser simuliert worden wären. Nicht nur weil sie für A & I gut geeignet sind, sondern weil sie allgemein sehr gut und stabil bezüglich dem Degenerieren sind. Genau nachgeprüft kann dies natürlich nur mit den echten Add Health Daten. Ein weiterer möglicher Schritt in diese Richtung wäre es einen Zusammenhang zwischen dem alpha- und beta- Wert sowie den Statistiken zu überprüfen. Dies könnte eventuell auch auf analytischem Wege funktionieren.

Anhang

A Quellcode

A.1 Attract and Introduce Generator

```
#Generates a network by using Attract and Introduce algorithm of Fowler
#Generator written by Arlind Nocaj
generateAttract<-function(alphaV=0.9,betaV=0.3,numNodes=750,
numEdges=numNodes*4.2,direct=TRUE)
{
  #rm(list=ls())# delete all previous variables
  library(igraph) # igraph packge needed for working with graphs
  n<-numNodes # number of nodes
  e<-numEdges # number of edges
  m<-e/n # average number of edges per node (degree)
  alpha<-alphaV
  beta<-betaV

# create first set of individual traits
  d1<-as.numeric(runif(n)<alpha)*runif(n) # distribution of attractiveness
  cl1<-as.numeric(runif(n)<beta) # distribution of prob. of introducing
friends
# create original network
  g1<-graph.empty(n,directed=direct) # create empty graph of size n
  while(ecount(g1)<e) { # loop until enough edges are produced
    dyad<-sample(n,2) # choose a random pair
    if(runif(1)<d1[dyad[2]]) { # check if person 1 nominates person 2 as a
friend
      if(runif(1)<cl1[dyad[1]]) { # check if person 1 introduces person 2
to friends
        ne<-neighbors(g1,dyad[1]-1,mode="out") # get person 1ÃŒs neighbors
        if(length(ne)>0) { # check if there is at least one existing
neighbor
          for(i in 1:length(ne)) { # loop through neighbors
            if(runif(1)<d1[ne[i]+1]) g1<-add.edges(g1,c(dyad[2]-1,ne[i])) #
introduce friends
            if(runif(1)<d1[dyad[2]]) g1<-add.edges(g1,c(ne[i],dyad[2]-1)) #
and add edges if
# they become friends
          }
        }
      }
      g1<-add.edges(g1,dyad-1) # add random pair edge at end
    }
    g1<-simplify(g1) # remove duplicate edges and loops
  }
  #Convert from igraph to a network (just a different format)
  m1<-get.adjacency(g1)
  detach(package:igraph)
  require(network)
  net<-network(m1,directed=direct)
  #store attract and introduce as node attributes
  net %v% "attract"<- d1
  net %v% "introduce"<-cl1

  net
}

```

A.2 Statistik - itriangle

\changestat.c

```
#include "changestats.h"

/* calculate changestatistics for itriangle */
CHANGESTAT_FN(d_itriangle) {
  Edge e;
  Vertex h, t, change, node3;
  int i;

  ZERO_ALL_CHANGESTATS(i);
  FOR_EACH_TOGGLE(i) {
    h = heads[i];
    t = tails[i];
    change = 0.0;
    CHANGE_STAT[0] = 0.0;
    if(N_INPUT_PARAMS>0){ /* match introduce attribute on third node
    */
        STEP_THROUGH_OUTEDGES(t, e, node3) { /* step through outedges
        of tail */
          if(INPUT_PARAM[node3]>0){/* third node needs Introduce
          attribute */
            if (DIRECTED) change += IS_OUTEDGE(node3, h) +
            IS_INEDGE(node3, h);
            else change += IS_UNDIRECTED_EDGE(node3,h);
          }
        }
        STEP_THROUGH_INEDGES(t, e, node3) { /* step through inedges
        of tail */
          if(INPUT_PARAM[node3]>0){/* third node needs Introduce
          attribute */
          if (DIRECTED) change += IS_OUTEDGE(node3, h) +
          IS_INEDGE(node3, h);
            else change += IS_UNDIRECTED_EDGE(node3,h);
          }
        }
      }

    TOGGLE_IF_MORE_TO_COME(i);
  }
  UNDO_PREVIOUS_TOGGLES(i);
}
```

\changestat.h

```
#ifndef CHANGESTATS_H
#define CHANGESTATS_H

#include "edgetree.h"

typedef struct ModelTermstruct {
  void (*func)(int, Vertex*, Vertex*, struct ModelTermstruct*,
  Network*);
  double *attrib; /* Ptr to vector of covariates (if necessary;
  generally unused) */
  int nstats;   /* Number of change statistics to be returned */
  double *dstats; /* ptr to change statistics returned */
  int ninputparams; /* Number of input parameters passed to function
  */
  double *inputparams; /* ptr to input parameters passed */
} ModelTerm;
.
.
.
CHANGESTAT_FN(d_itriangle);

#endif
```

\InitErgmTerm.users.R

```
#itriangle (introduce triangle) statistic
InitErgmTerm.itriangle <- function(nw, arglist, ...) {
    a <- check.ErgmTerm(nw, arglist, directed=NULL, bipartite=NULL,
                        varnames = c("attrname"),
                        vartypes = c("character"),
                        defaultvalues = list(NULL),
                        required = c(TRUE))
  # Process the arguments
  nodecov <- get.node.attr(nw, a$attrname)


  # Construct the output list
  list(name="itriangle",             #name: required
       coef.names = paste("itriangle", a$attrname, sep="."),#coef.names:
required
       inputs = nodecov,             #There is just one  covariate
       soname = "ergmuserterms",  # So "ergm" knows where to find it!
       dependence = TRUE # So we don't use MCMC if not necessary
       )
}
```

A.3 Simulationsfunktion

```
#generates networks with attract and introduce method. Density is the same es used by fowler in his Paper.
#node amount can be set
#each A & I Network is being fit and afterwards graphs are being generated from the resulting model.
#2 Boxplots will be drawn. One show the distribution of the original A & I networks, the other show the distribution of the Simulated ERGM networks
# if the boxplots are pretty close it seems like the attract and introduce can be modelled pretty good with the ergm
testFowler2<-function(numNodes=10,numEdges=numNodes*4.2,amount=10, fname="standardname"){
  library(ergm)
  size<-numNodes
  #used statistics
  statistics=c("dist","ideg","odeg","espart","dspart")
  #statistics=c("dist","deg","espart","dspart")
  config=vector("list",3)
  config[["statistics"]]=statistics
  #matrix of observed networks
  observed=list()
  #matrix of means of simulated networks
  simulated=list()
  config[["observed"]]=observed
  config[["simulated"]]=simulated

  #save the result so that it can be plottet also in another way
  filename<-paste(fname,"NetworkSize",size,"NetNr.",amount,sep="-")

  for (netNumber in c(1:amount)){
    tryCatch({
          name<-paste("NetworkSize",size,"NetNr.",netNumber,sep="-")
          print(name)
          network<-generateAttract(0.9,0.3,numNodes=size, direct=TRUE)#generate Network with Attract and Introduce
          model1<-ergm(network~edges+gwesp+gwdsp+nodecov("attract"), Verbose=TRUE)

          print("Fitting done.")
          fit<-gof(model1,nsim=50,GOF=~distance+degree+espartners+ dspartners)
          print("Goodness of Fit done.")
          config<-addToMatrix(config,fit,statistics)
          #print(fit)
        }, error = function(ex) {
          cat("xxxxxx  Fit didn't work. \n", sep="")
        }, finally = {

        })

  }

  save(config,file=filename)
  return(config)
}

```

```
 1: # function used for storing the results of the simulations
 2: addToMatrix<-function(config,fit,statistics){
 3:   for (stat in statistics){
 4:     table<-fit[[paste("summary.",stat,sep="")]]
 5:     if (is.null(config[["observed"]][[stat]])){
 6:       config[["observed"]][[stat]]=table[,1]
 7:       config[["simulated"]][[stat]]=table[,3]
 8:     }else{
 9:       config[["observed"]][[stat]]=cbind(config[["observed"]][[stat]],
    table[,1])
10:       config[["simulated"]][[stat]]=cbind(config[["simulated"]][[stat]],
    table[,2])
11:       config[["simulated"]][[stat]]=cbind(config[["simulated"]][[stat]],
    table[,3])
12:       config[["simulated"]][[stat]]=cbind(config[["simulated"]][[stat]],
    table[,4])
13:     }
14:   }
15:   return(config)
16: }
```

A.4 Plotfunktion

```
#function for plotting general Test Results
plotX<-function(config,pdffile=NULL,logy=FALSE){
  gstat=list()
  gstat[["dist"]]=c("minimum geodesic distance","proportion of dyads")
  gstat[["ideg"]]=c("in degree","proportion of nodes")
  gstat[["odeg"]]=c("out degree","proportion of nodes")
  gstat[["espart"]]=c("edge-wise shared partners","proportion of edges")
  gstat[["dspart"]]=c("dyad-wise shared partners","proportion of edges")

  if (is.null(pdffile)){
    x11()
  }else {
    pdf(pdffile)
  }
  x11()
#first the observed statistics need to be plotted so that we get the values
  where to draw the line
  par(mfrow=c(4 ,2))
  boxes=list()
  for (stat in config[["statistics"]]){
    #clean data
    goodrows=c()

    data1=config[["observed"]][[stat]]
    data2=config[["simulated"]][[stat]]

    for (r in c(1:nrow(data1))){
      if (!(sum(data1[r,])==0 && sum(data2[r,])==0)){
        goodrows=c(goodrows,r)
      }
    }
    data1=data1[goodrows,]
    data2=data2[goodrows,]

    data1=t(data1)
    if (logy){
      data1=log(data1)
    }
    boxes[[stat]]=boxplot(data1,col="BLUE")
  }

  dev.set(dev.prev())
  for (stat in config[["statistics"]]){
    #clean data
    goodrows=c()

    data1=config[["observed"]][[stat]]
    data2=config[["simulated"]][[stat]]

    for (r in c(1:nrow(data1))){
      if (!(sum(data1[r,])==0 && sum(data2[r,])==0)){
        goodrows=c(goodrows,r)
      }
    }
    data1=data1[goodrows,]
    data2=data2[goodrows,]

    data2=t(data2)
    if (logy){
      data2=log(data2)
```

```
60:     }
61:     ylab=gstat[[stat]][2]
62:     if (logy)
63:       ylab=paste("log - ",ylab,sep="")
64:     box=boxplot(data2,xlab=gstat[[stat]][1],ylab=ylab)
65:     b1=box$stats[1,]
66:     b2=box$stats[5,]
67:     hinge1=boxes[[stat]]$stats[1,]
68:     hinge2=boxes[[stat]]$stats[5,]
69:     lines(x=1:c(length(hinge1)),y=hinge1,col="GREEN")
70:     lines(x=1:c(length(hinge2)),y=hinge2,col="GREEN")
71:     #lines(x=1:c(length(b1)),y=b1,col="GREY")
72:     #lines(x=1:c(length(b2)),y=b2,col="GREY")
73:   }
74:   if (!is.null(pdffile)){
75:     graphics.off()
76:   }
77: }
```

B Goodness of Fit - spezieller Test

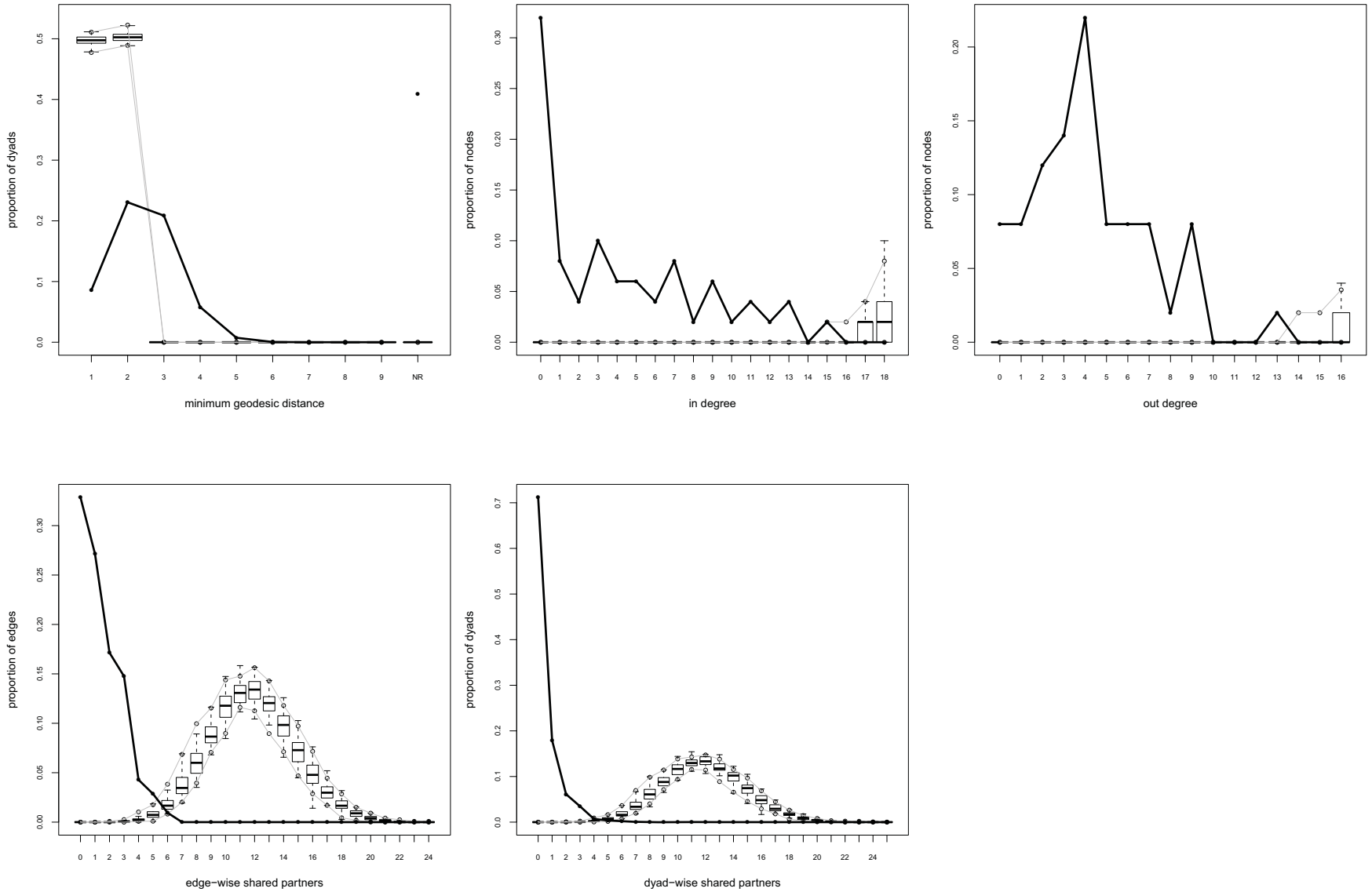


Abbildung 12: (1): edges + nodeicov

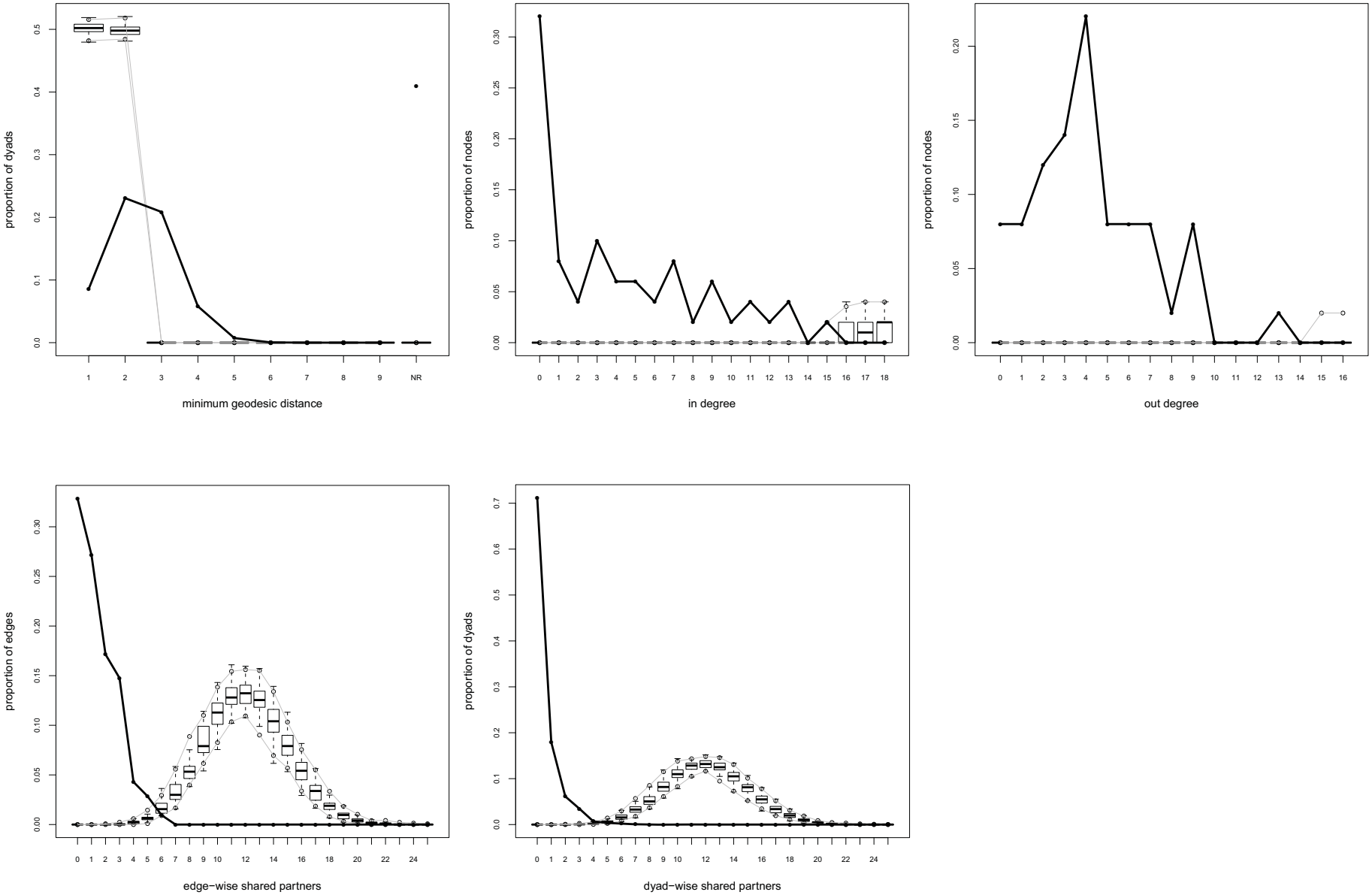


Abbildung 13: (2): edges + nodeicov + triadcensus

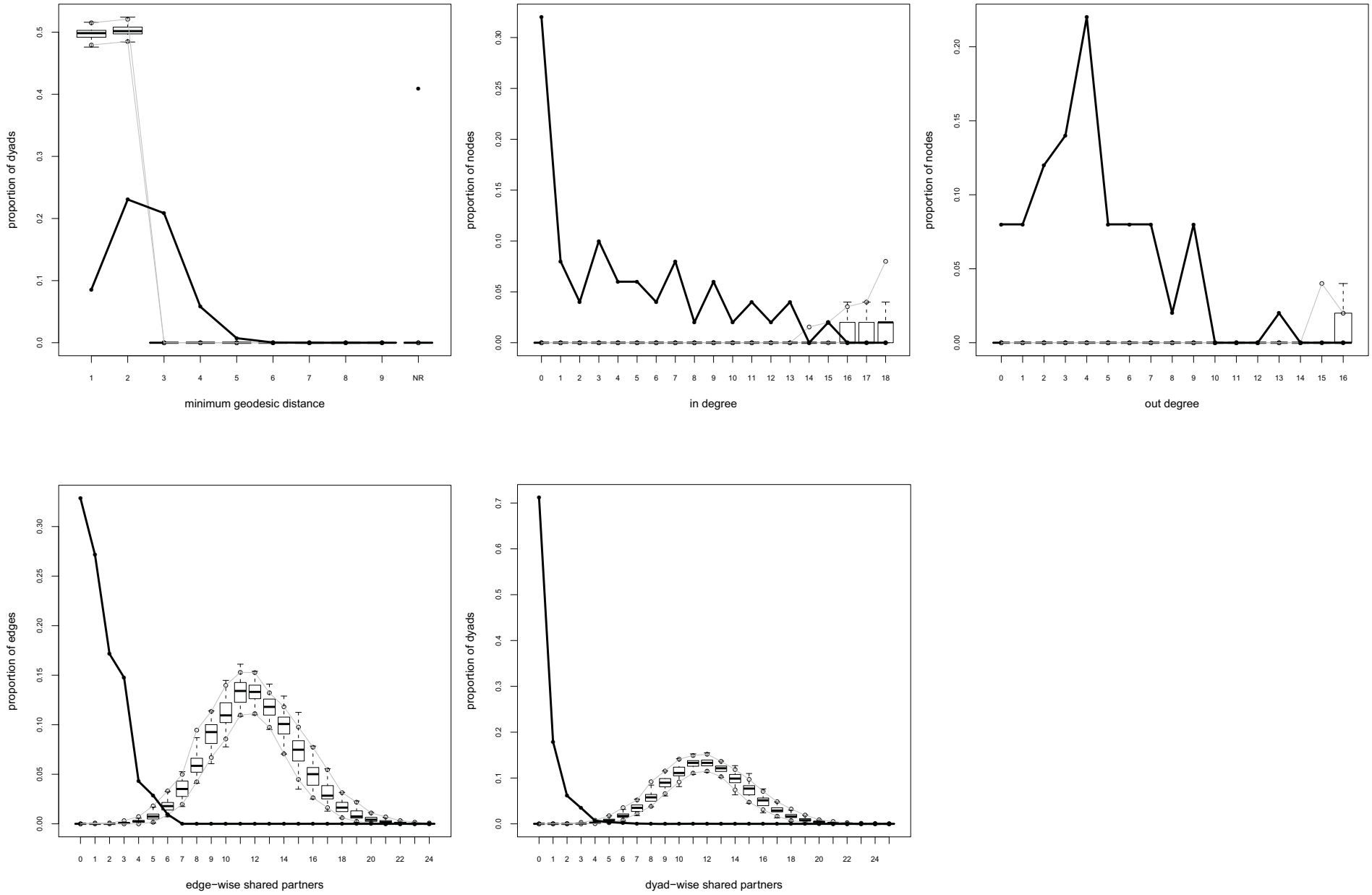


Abbildung 14: (3): edges + nodeicov + itriangle

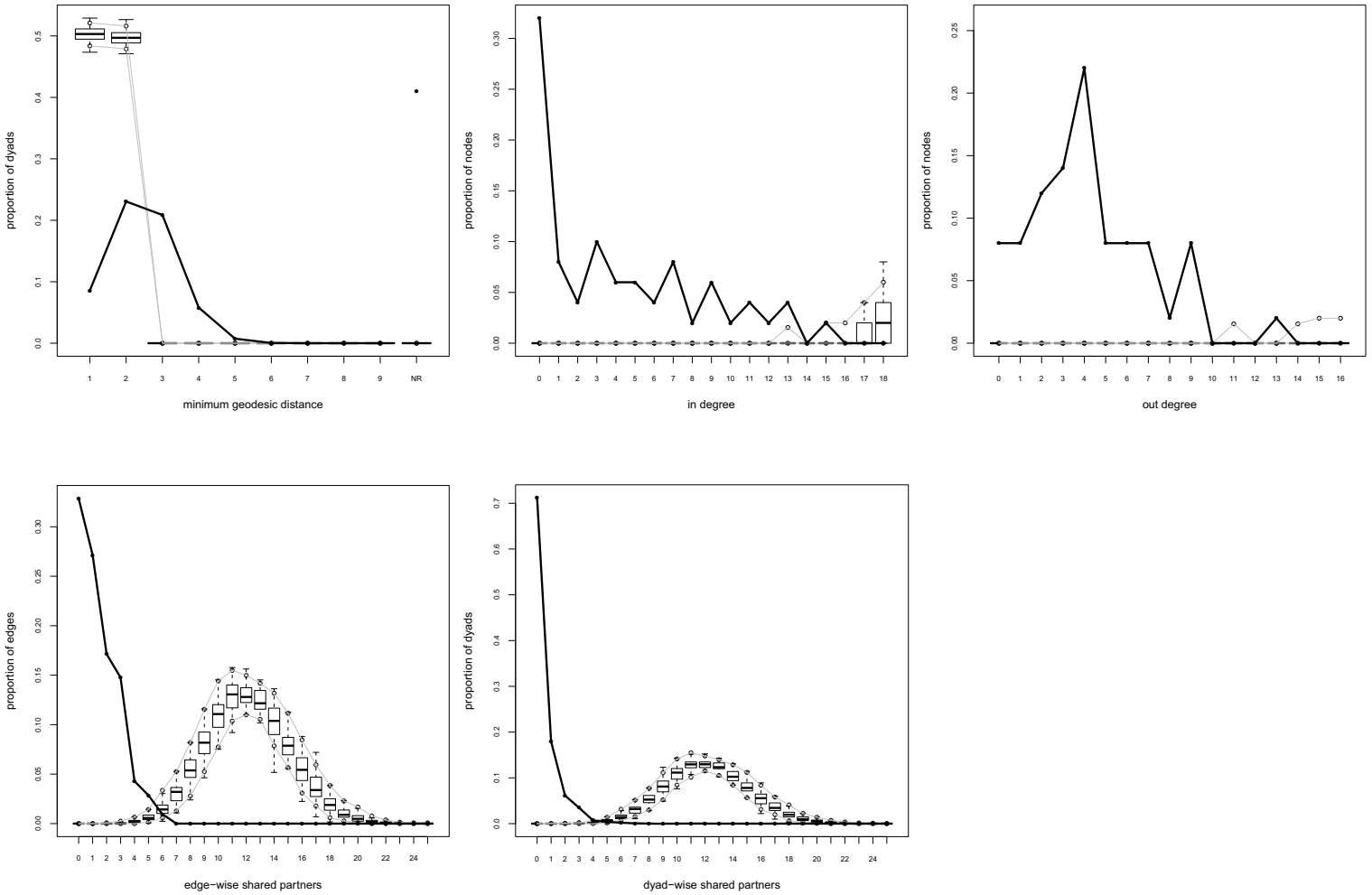


Abbildung 15: (4): edges + gwesp + gwdsp + gwidegree + gwodegree

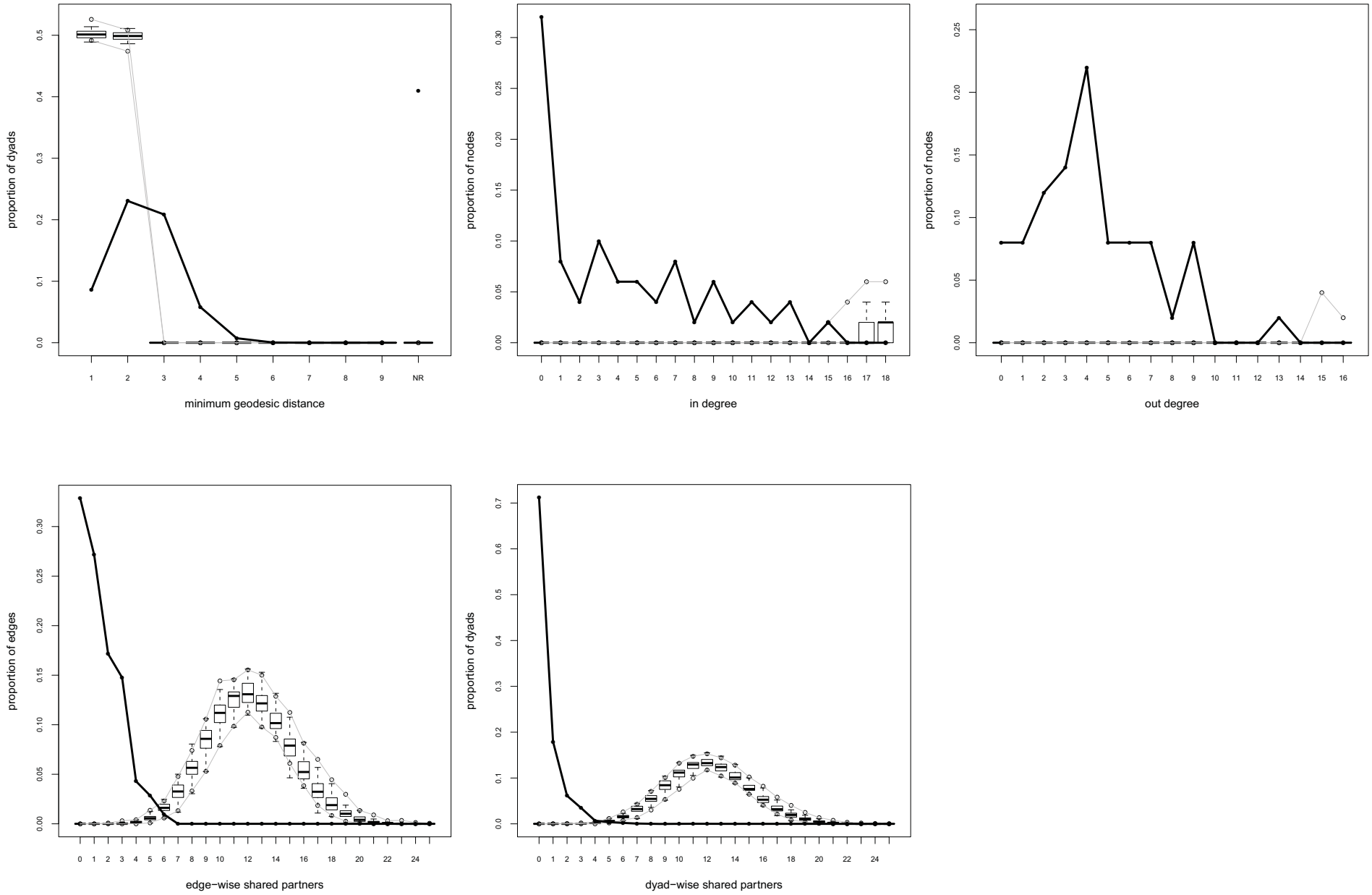


Abbildung 16: (5): edges + gwesp + gwdsp + gwidegree + gwodegree + nodeicov

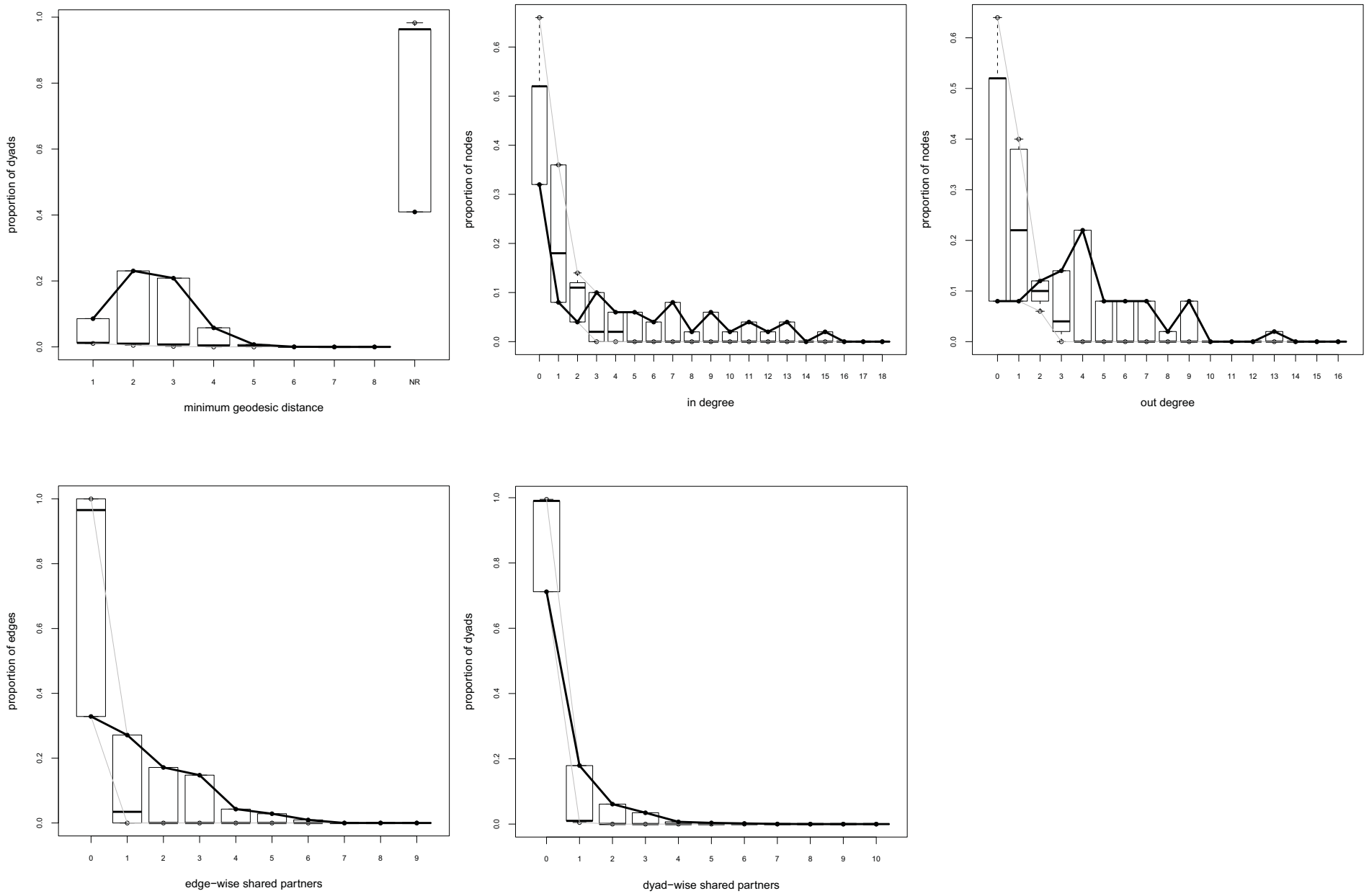


Abbildung 17: (6): edges + gwesp + gwdsp + nodeicov

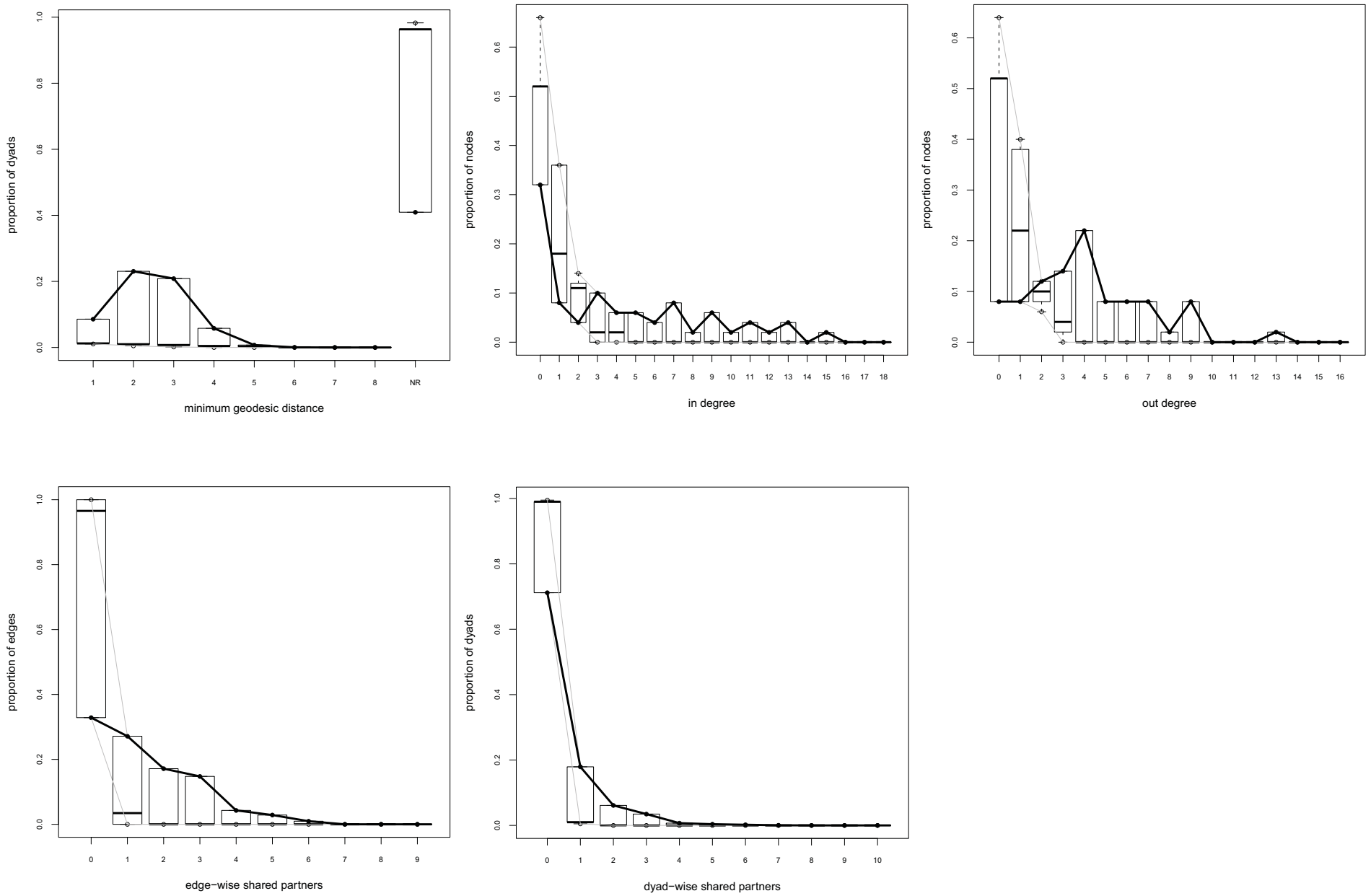


Abbildung 18: (7): edges + gwesp + gwdsp + nodeicov + itriangle

Goodness of Fit - spezieller Test - logarithmisch skaliert

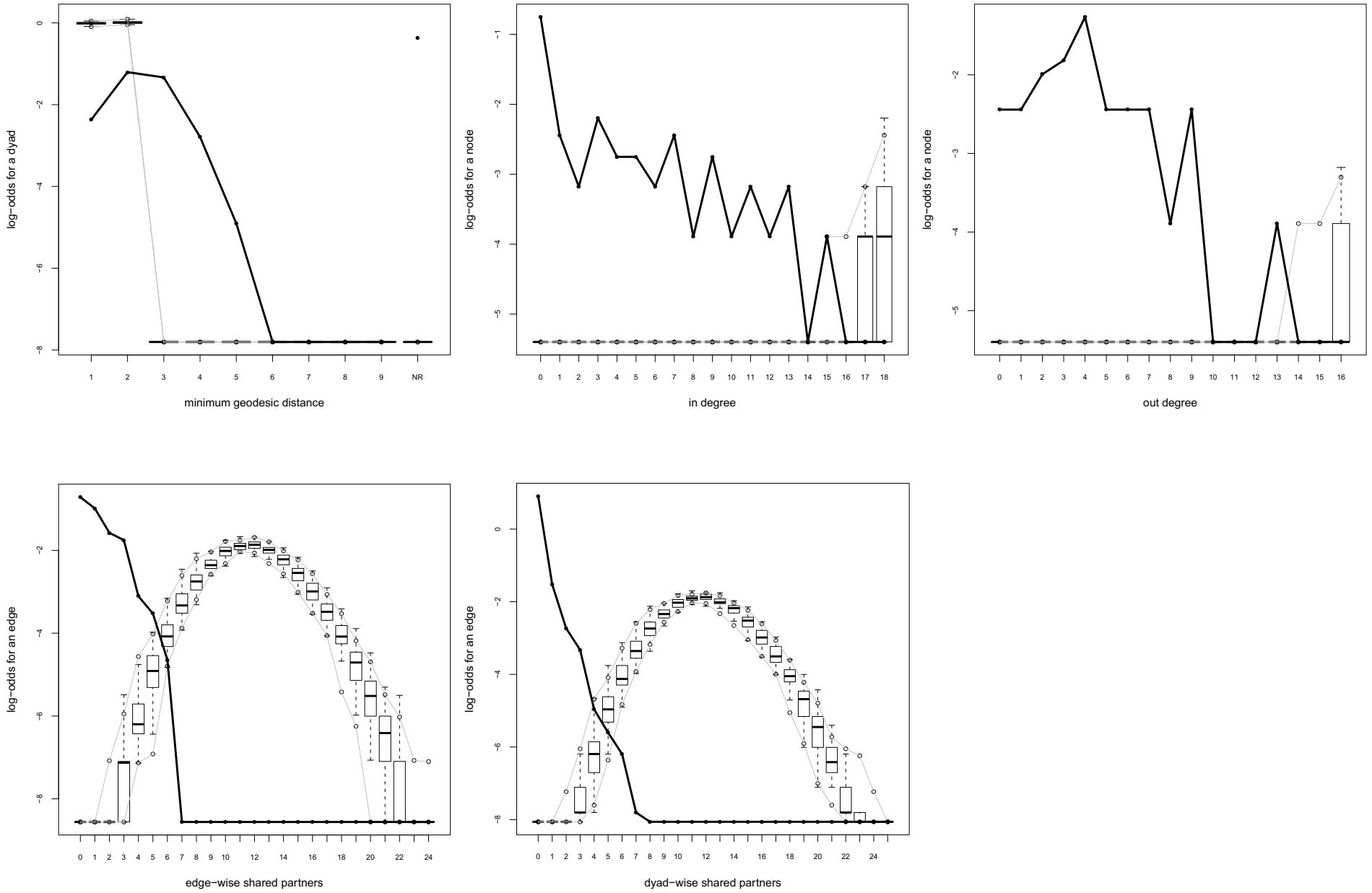


Abbildung 19: (1): edges + nodeicov

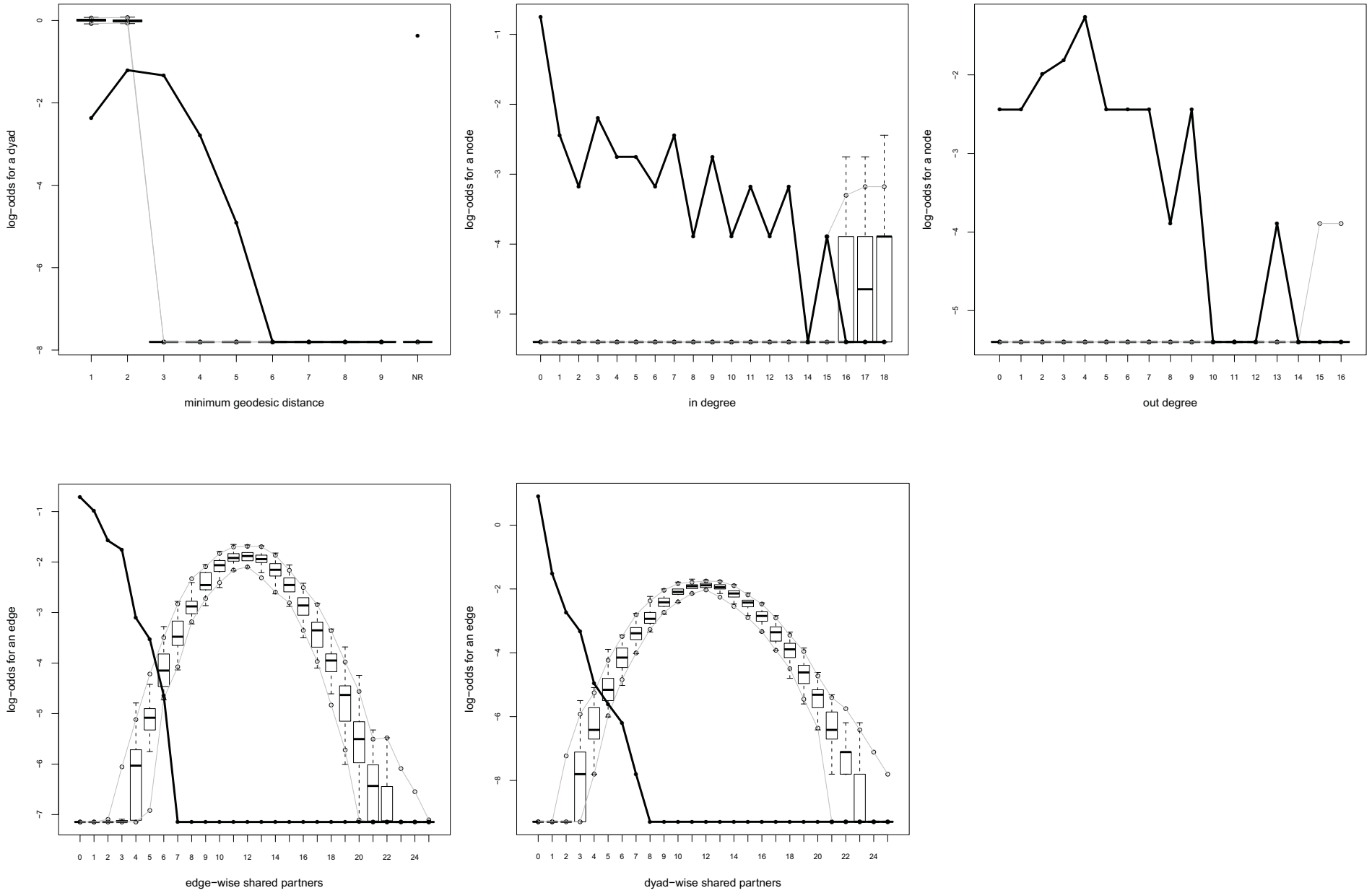


Abbildung 20: (2): edges + nodeicov + triadcensus

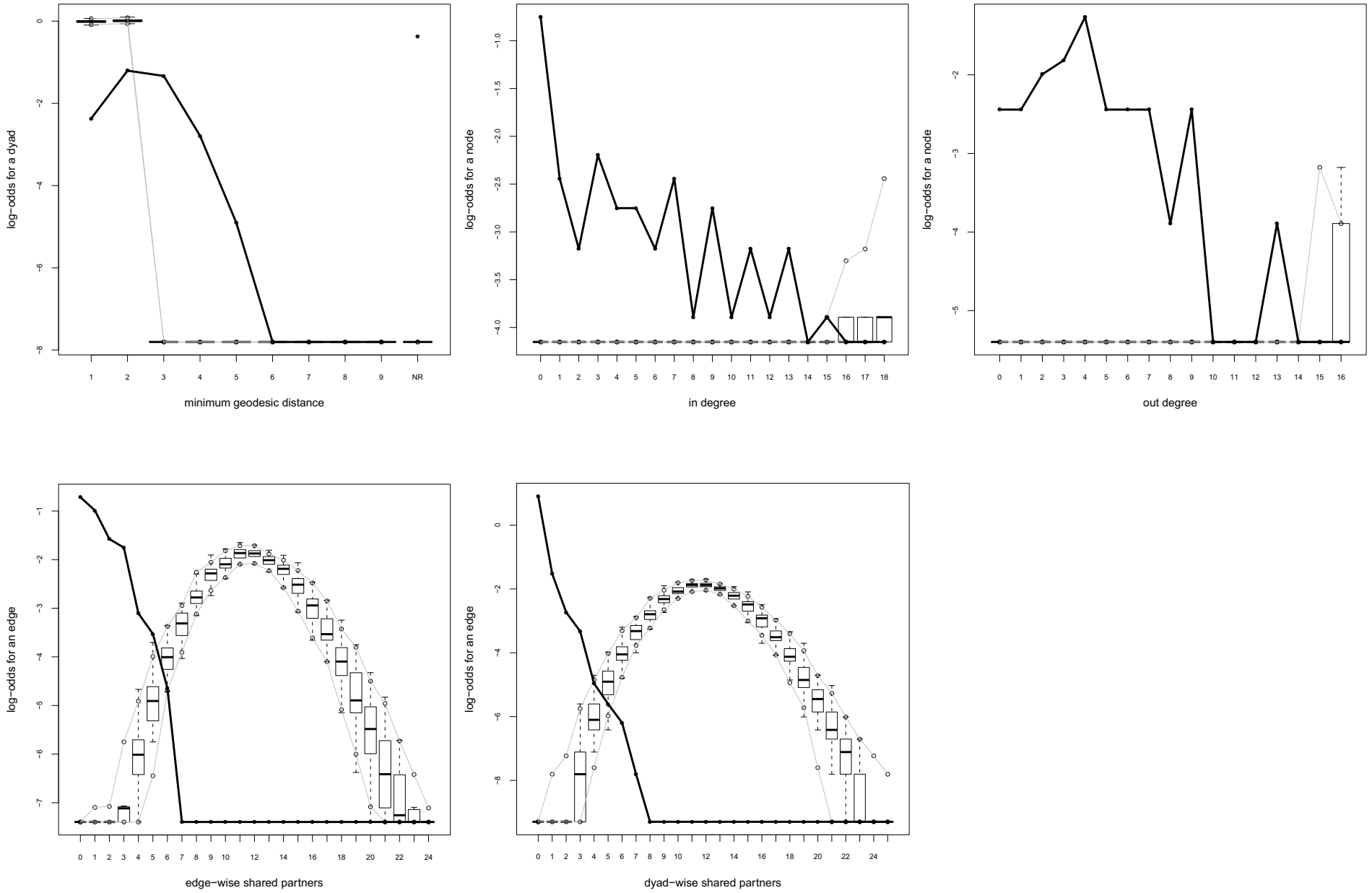


Abbildung 21: (3): edges + nodeicov + itriangle

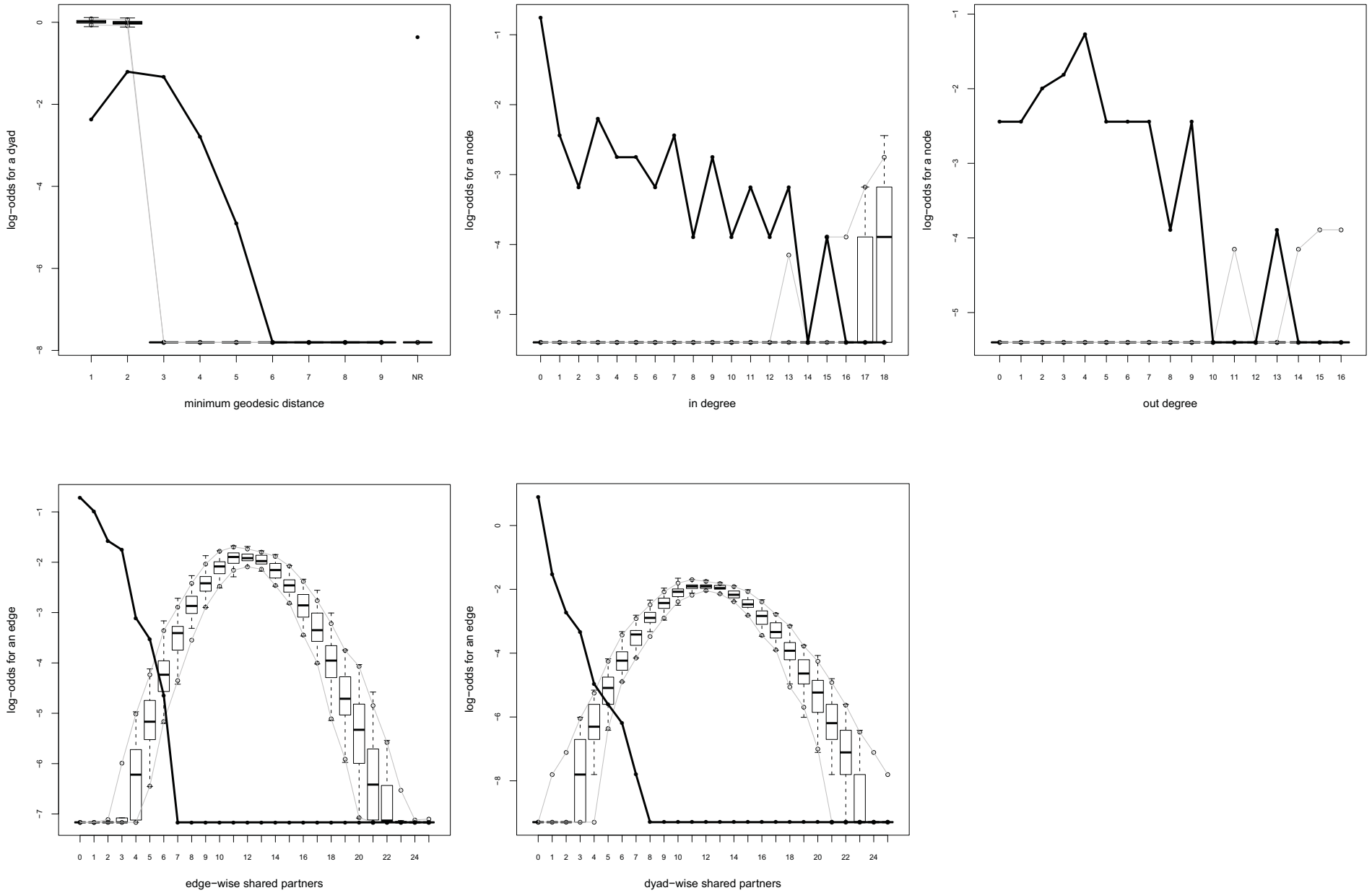


Abbildung 22: (4): edges + gwesp + gwdsp + gwidegree + gwodegree

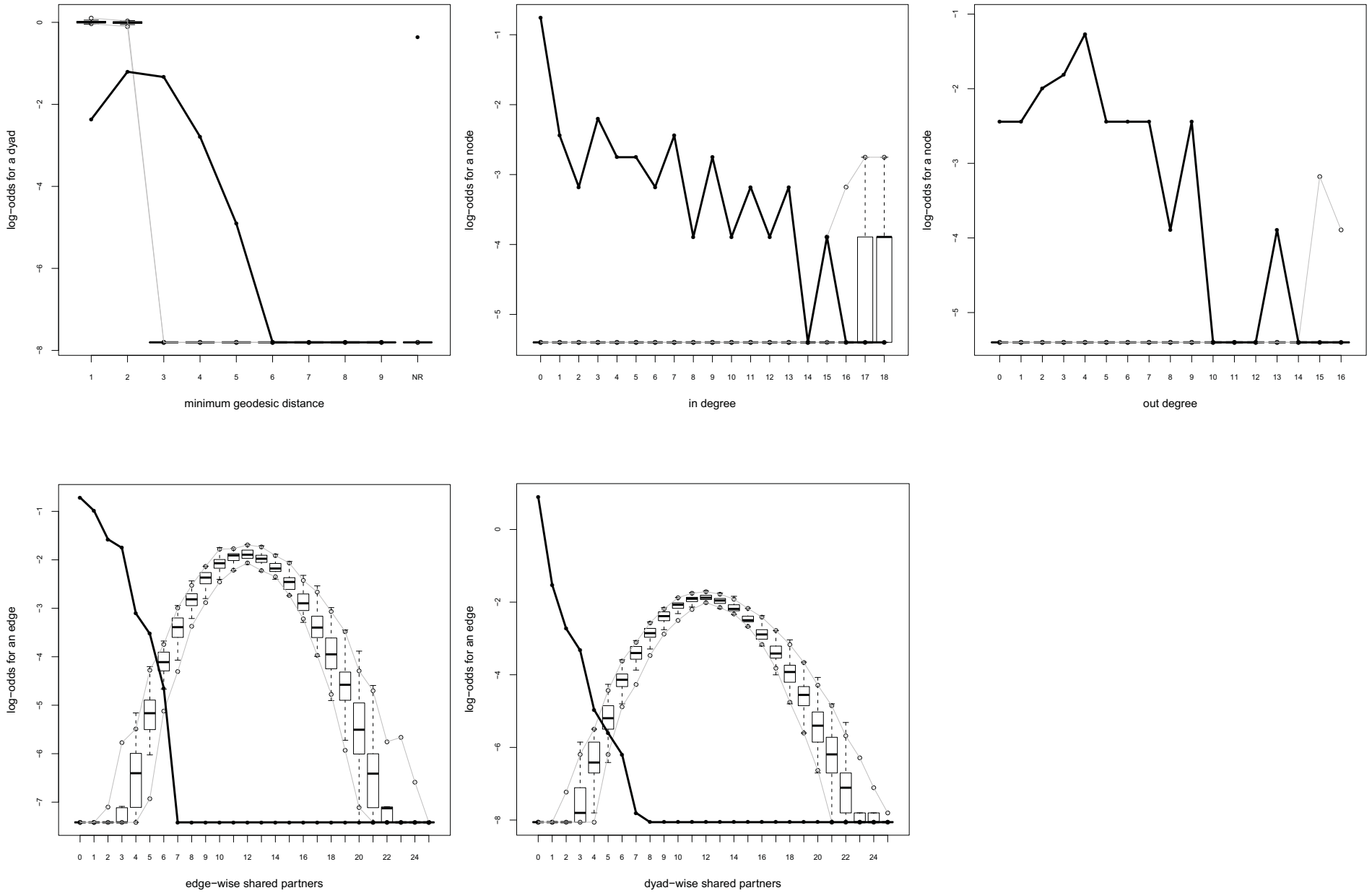


Abbildung 23: (5): edges + gwesp + gwdsp + gwidegree + gwodegree + nodeicov

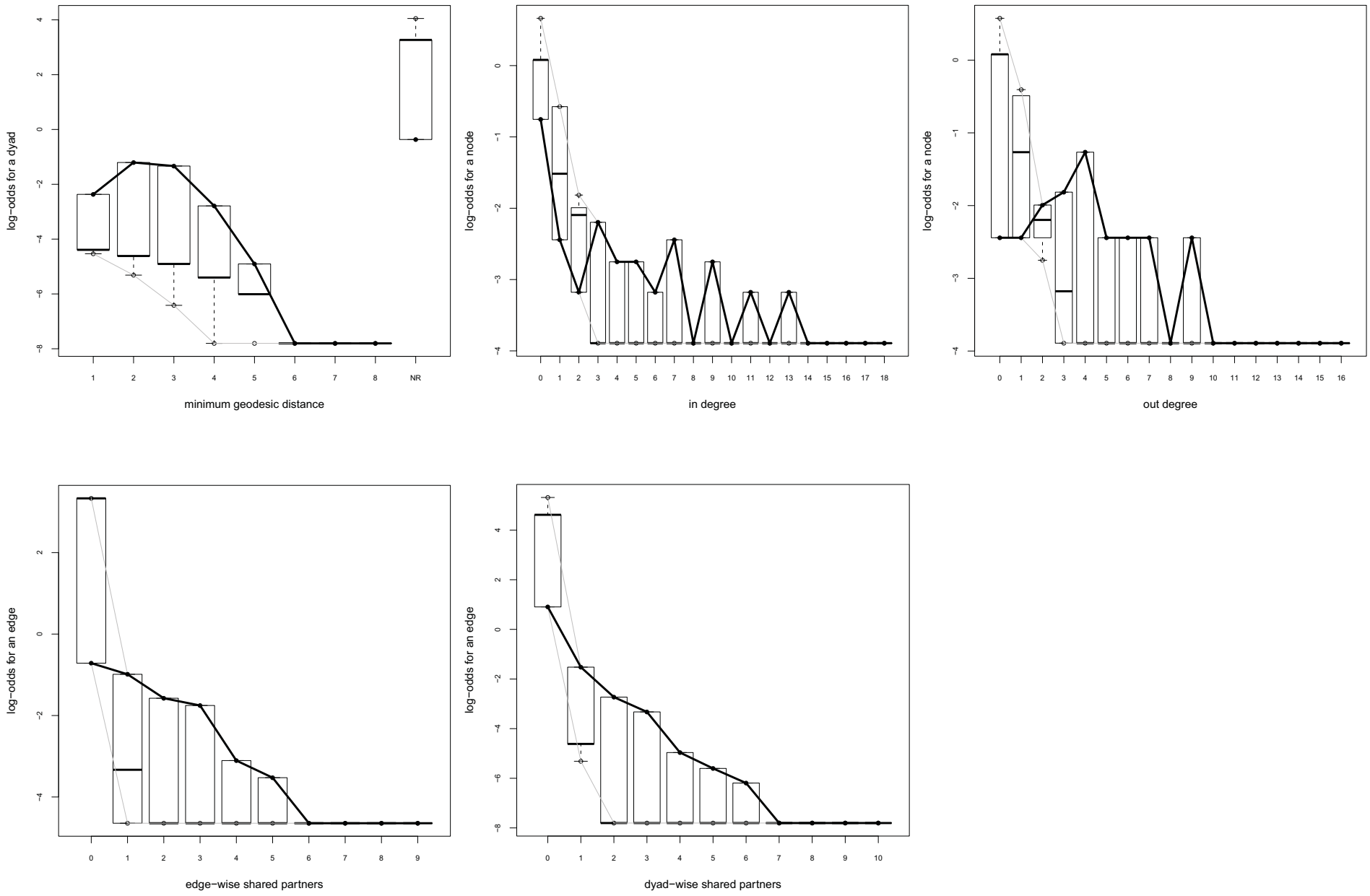


Abbildung 24: (6): edges + gwesp + gwdsp + nodeicov

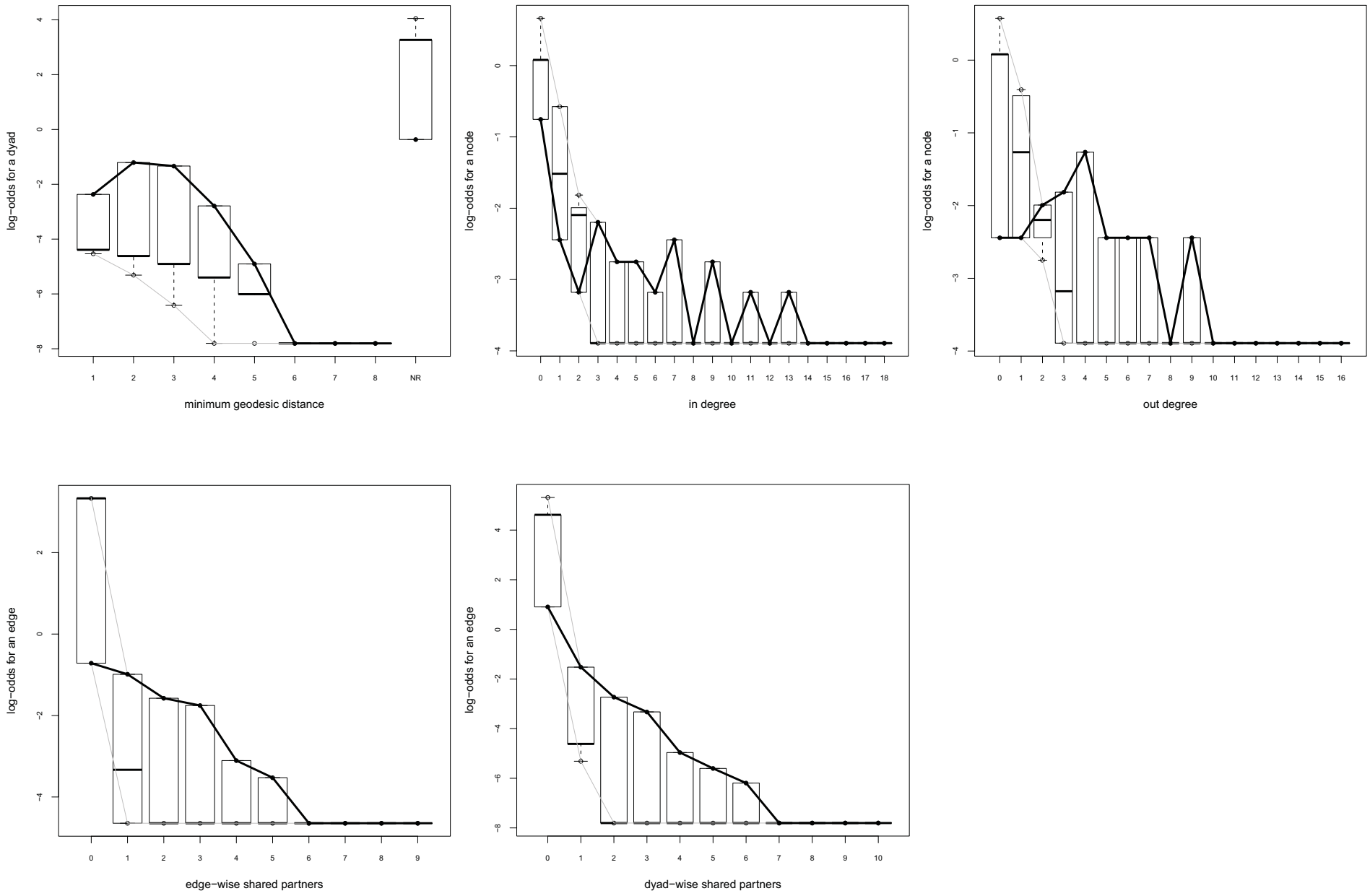


Abbildung 25: (7): edges + gwesp + gwdsp + nodeicov + itriangle

C Goodness of Fit - Genereller Test

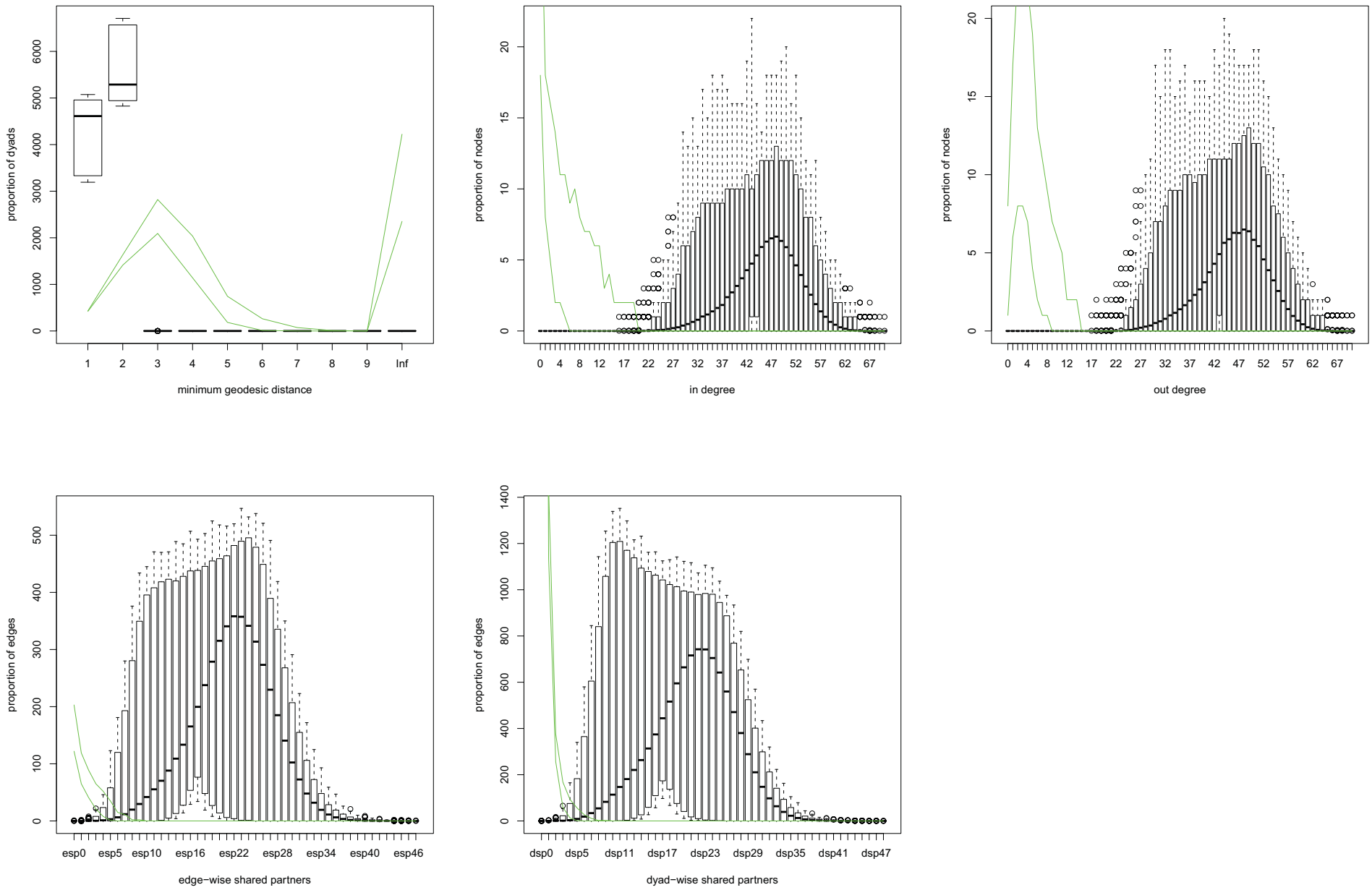


Abbildung 26: (1): edges + nodeicov

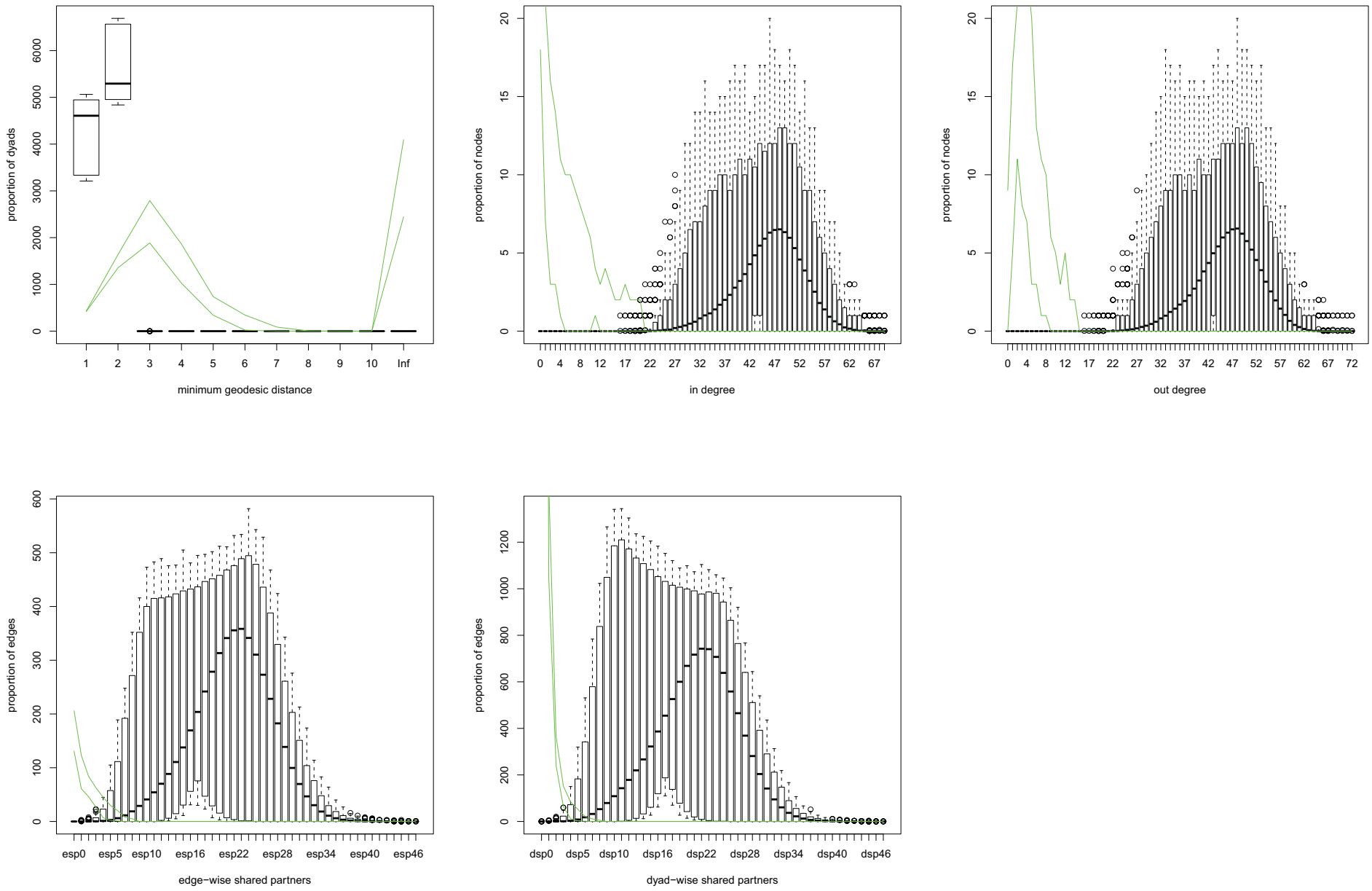


Abbildung 27: (2): edges + nodeicov + triadcensus

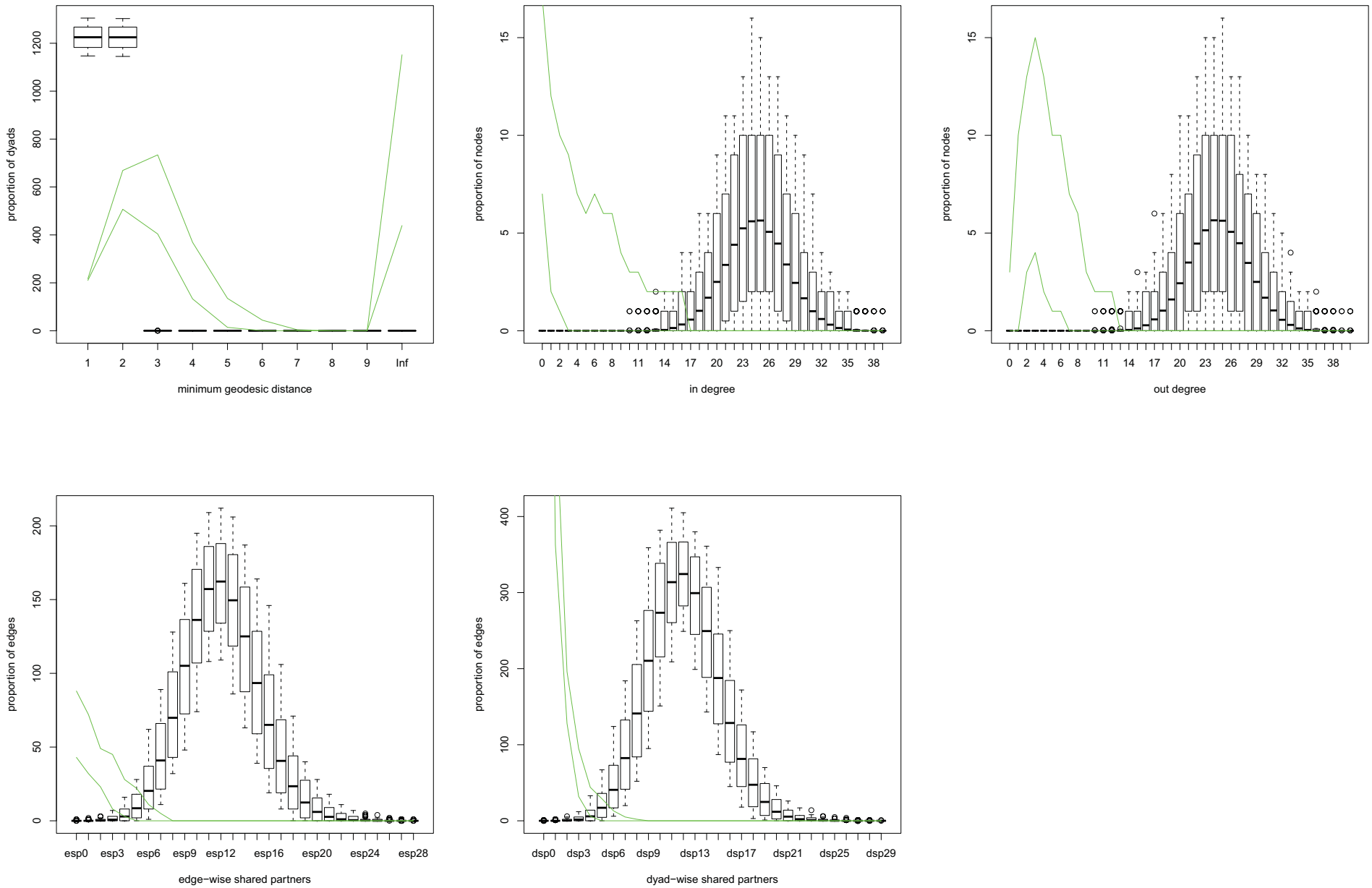


Abbildung 28: (4): edges + gwesp + gwdsp + gwidegree + gwodegree

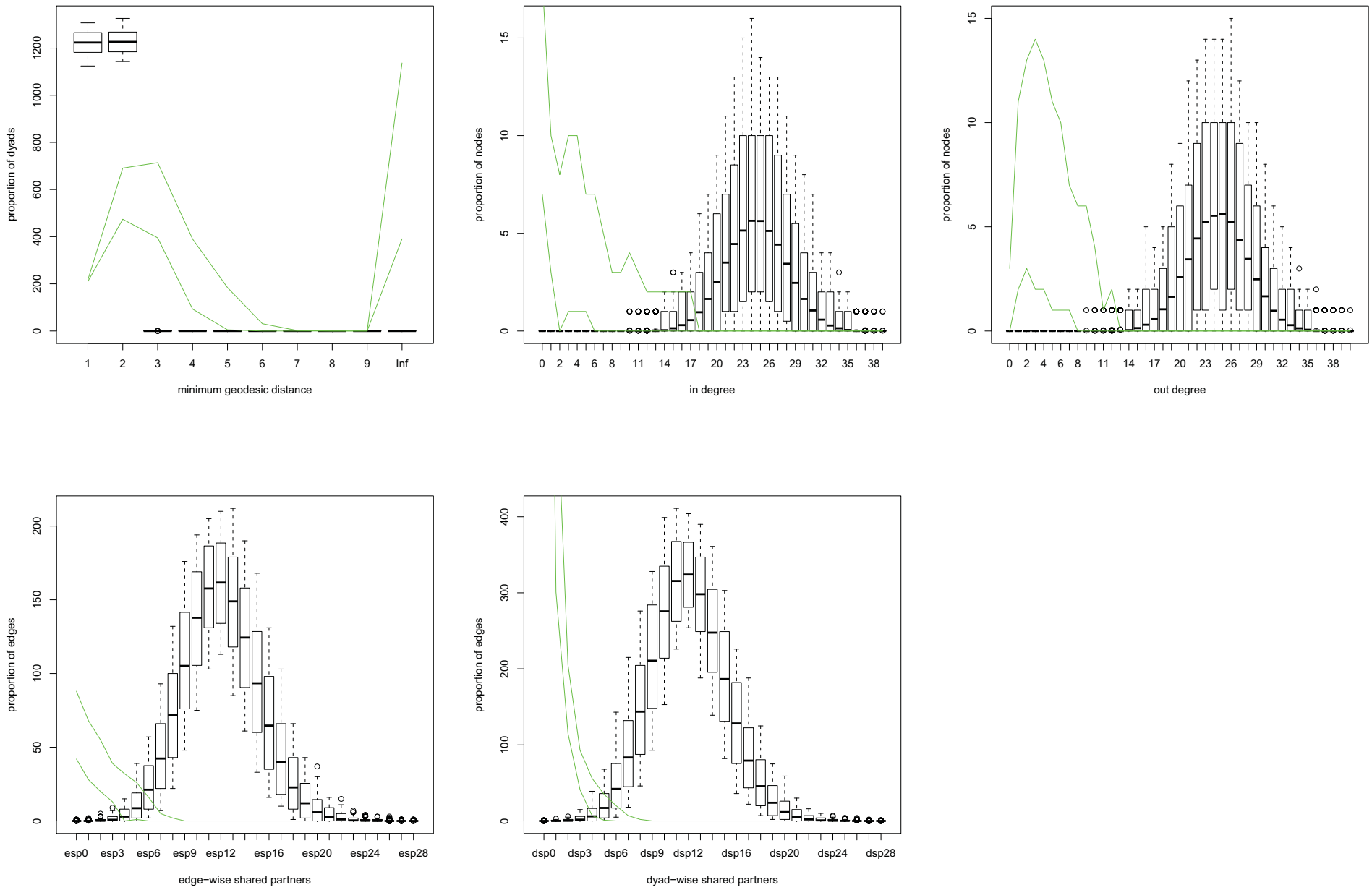


Abbildung 29: (5): edges + gwesp + gwdsp + gwidegree + gwodegree + nodeicov

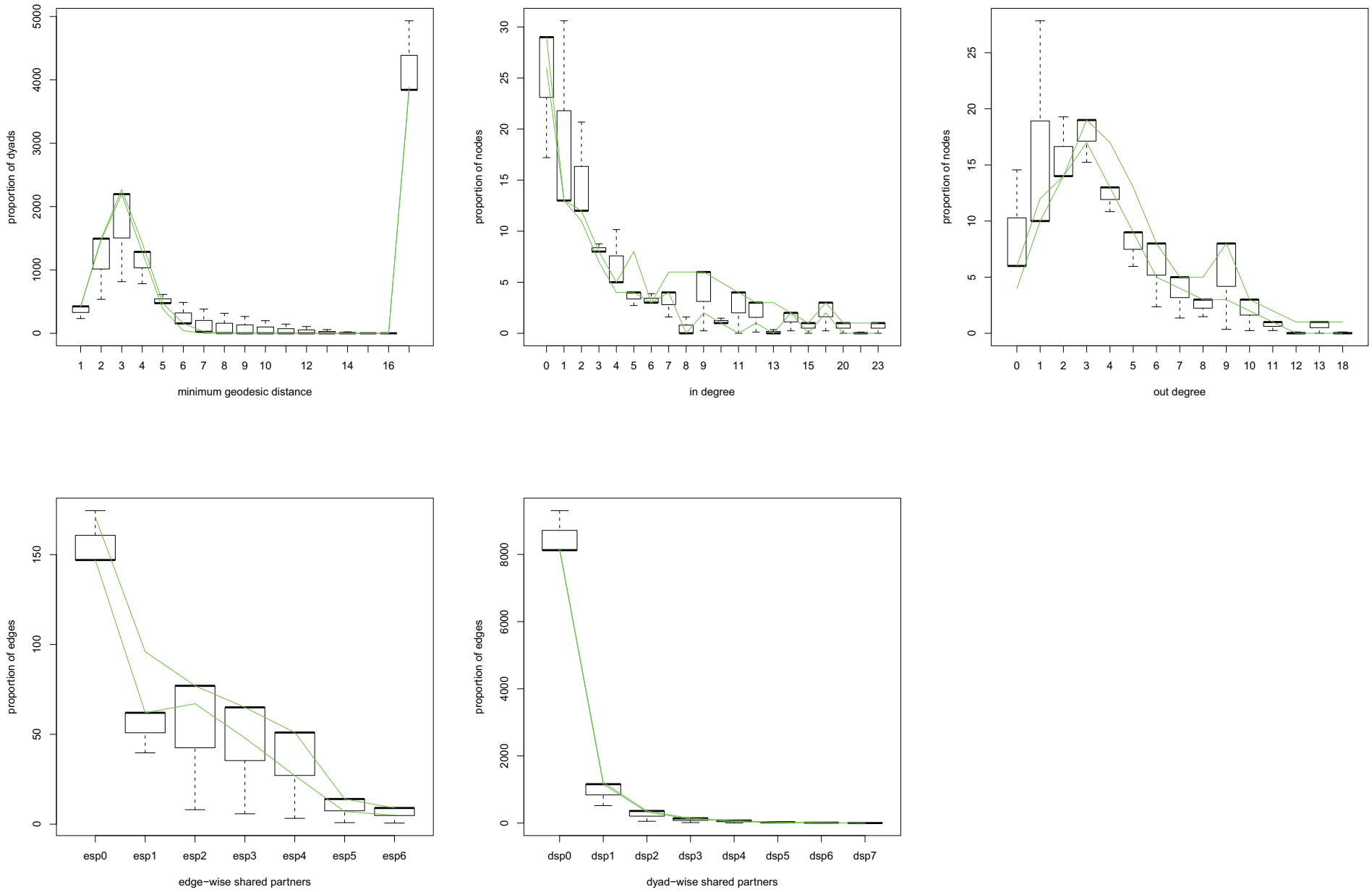


Abbildung 30: (6): edges + gwesp + gwdsp + nodeicov

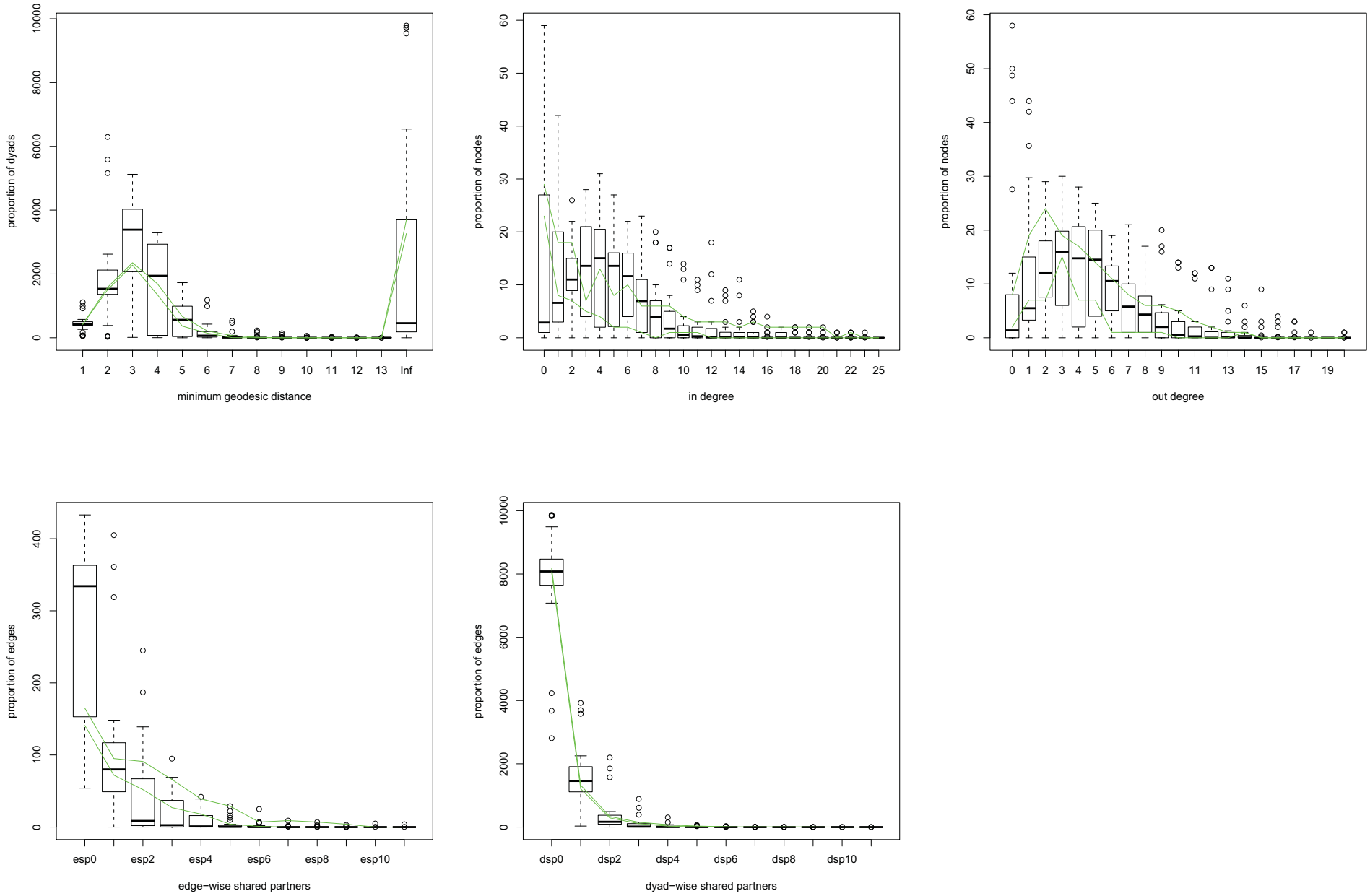
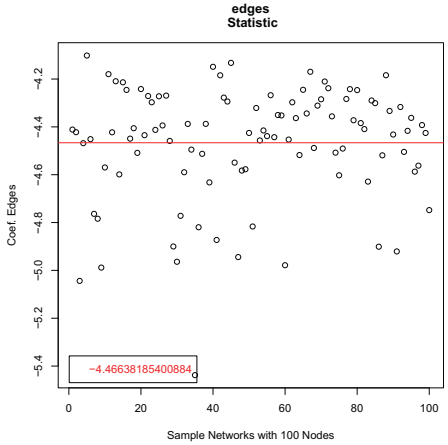
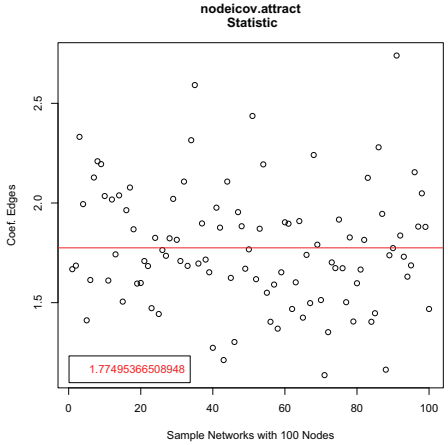


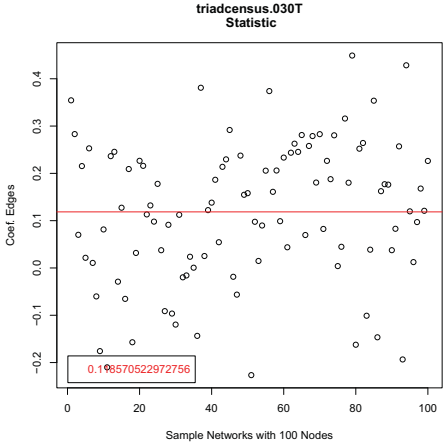
Abbildung 31: (7): edges + gwesp + gwdsp + nodeicov + itriangle

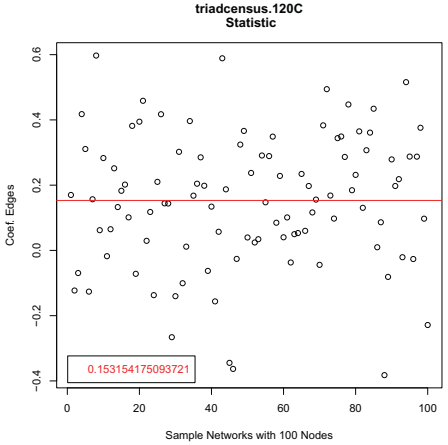
D Parameter für Spiegel Netzwerk

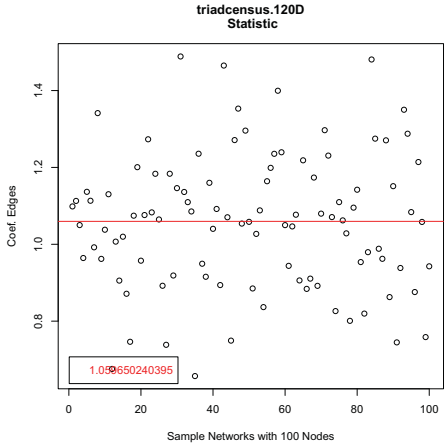
Parameter für (2), Knoten=100, inkl. edges

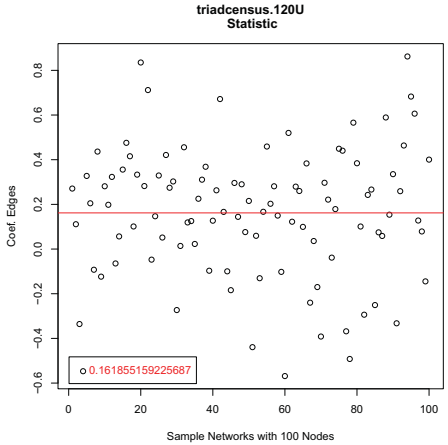


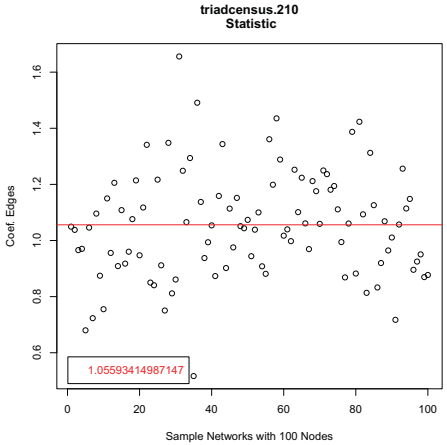


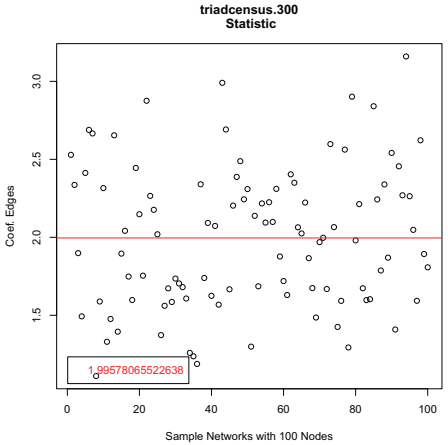




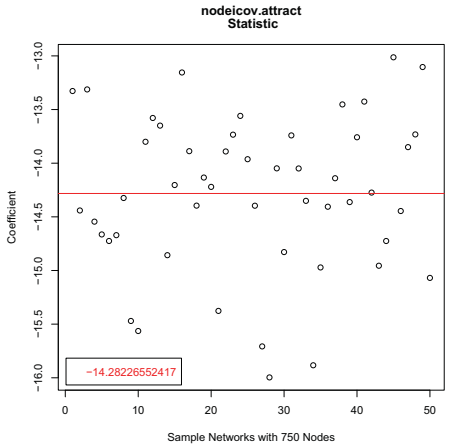


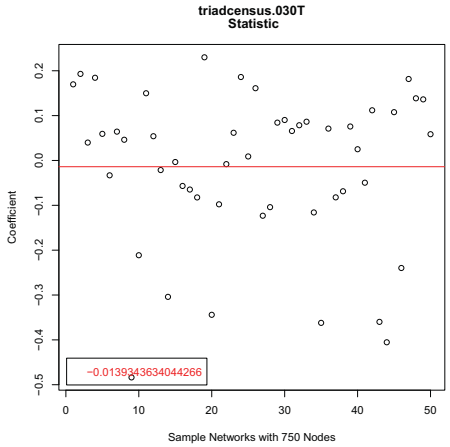


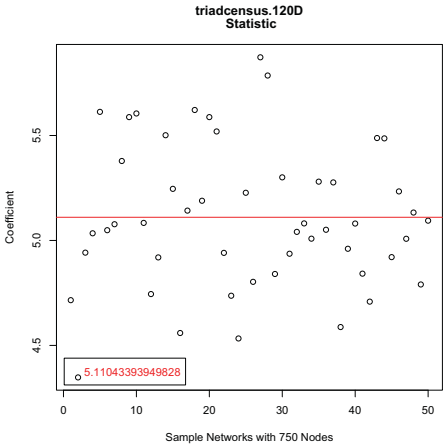


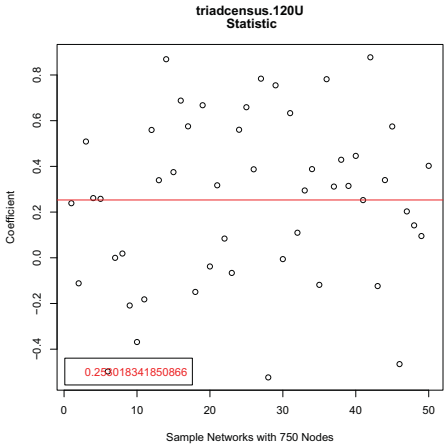


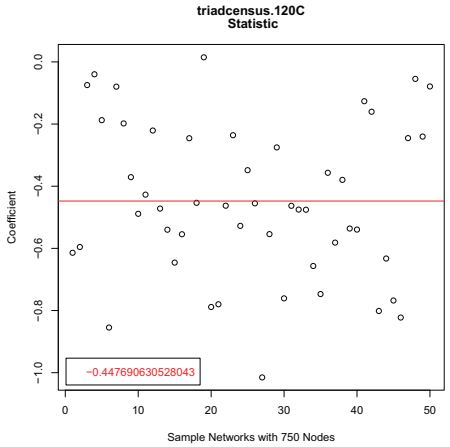
Parameter für (2), Knoten=100, ohne edges

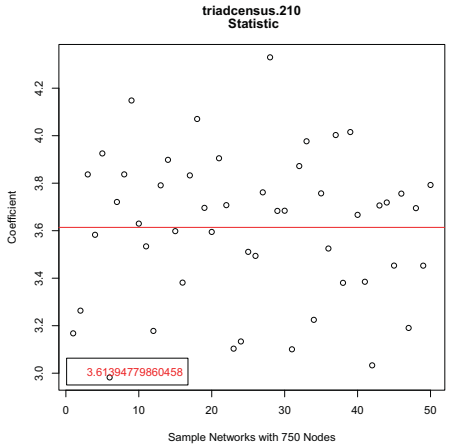


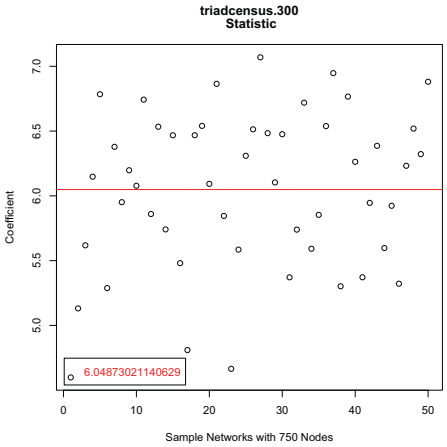




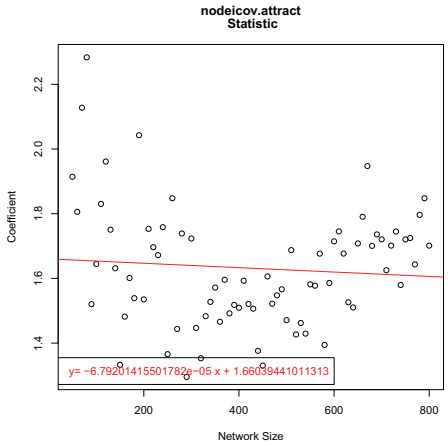


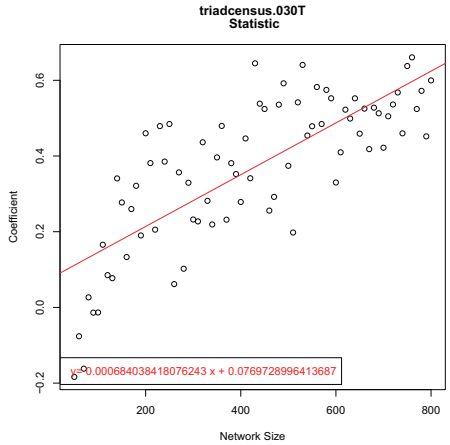


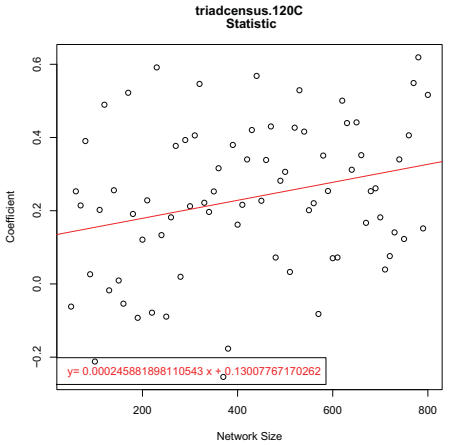


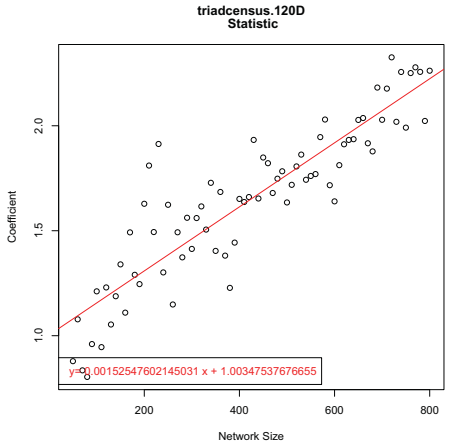


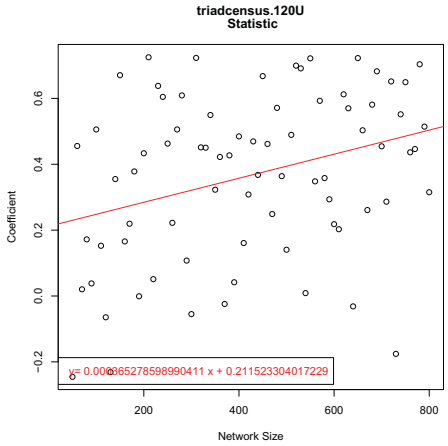
Parameter für (2), Knoten=50 - 800, ohne edges

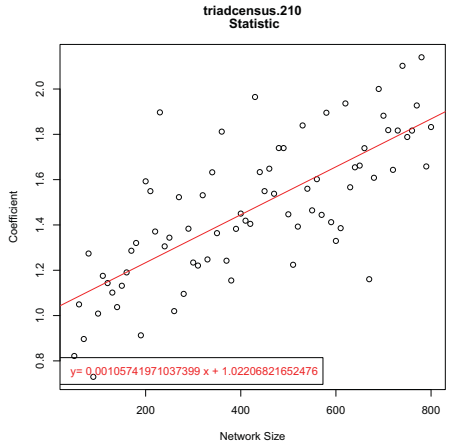


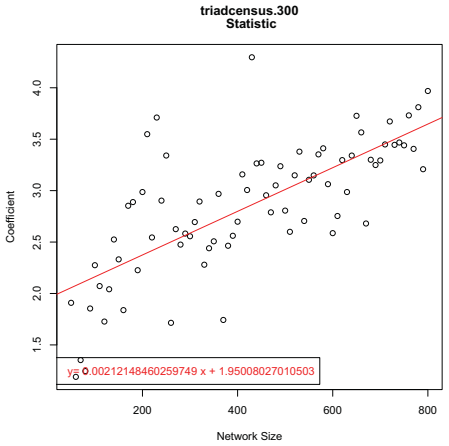




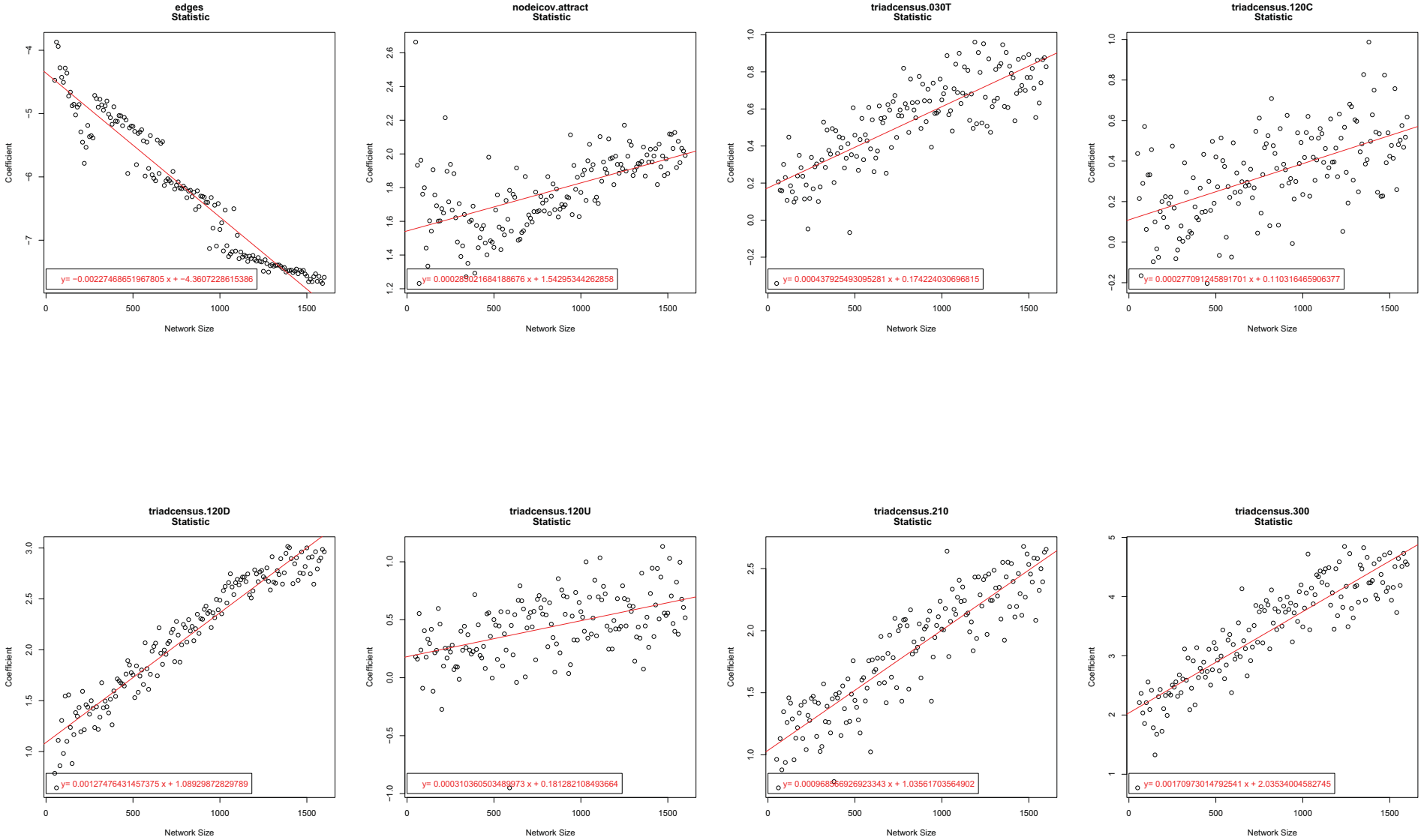








Parameter für (2), Knoten=50 - 1600, inkl. edges



Literatur

[1] J. H. Fowler, C. T. Dawes, and N. A. Christakis. Model of genetic variation in human social networks. *Proceedings of the National Academy of Sciences*, 106(6):1720–1724, February 2009.

[2] M. S. Handcock, D. R. Hunter, C. T. Butts, S. M. Goodreau, M. Morris, and P. Krivitsky. *ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks*. Seattle, WA, 2010. Version 2.2-2. Project home page at urlhttp://statnetproject.org.

[3] D. R. Hunter, S. M. Goodreau, and M. S. Handcock. Goodness of fit of social network models. *Journal of the American Statistical Association*, 103:248–258, 2008.

[4] A. Nocaj. Ergm parameter estimation. Seminarausarbeitung, Universität Konstanz, 2010.

[5] G. Robins, P. Pattison, Y. Kalish, and D. Lusher. An introduction to exponential random graph (p*) models for social networks. *Social Networks*, 29(2):173–191, May 2007.

[6] T. A. B. Snijders, P. E. Pattison, G. L. Robins, and M. S. Handcock. New specifications for exponential random graph models. 2004.