UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Metka Matkovič

Ocenjevanje in primerjava različnih metod za bločno modeliranje na simuliranih omrežjih

Magistrsko delo

Ljubljana, 2015

UNIVERZA V LJUBLJANI
FAKULTETA ZA DRUŽBENE VEDE

Metka Matkovič

Mentor: doc. dr. Aleš Žiberna

Ocenjevanje in primerjava različnih metod za bločno modeliranje na simuliranih omrežjih

Magistrsko delo

Ljubljana, 2015

Zahvaljujem se mentorju doc. dr. Alešu Žiberni, za pomoč in usmerjanje pri nastajanju dela.

Hvala družini in bližnjim za potrpežljivost in podporo tekom študija.

Ocenjevanje in primerjava različnih metod za bločno modeliranje na simuliranih omrežjih

Analiza socialnih omrežij je metoda, ki se ukvarja s preučevanjem odnosov v množici enot (Hlebec in Kogovšek 2006, 7–9). Eden izmed ciljev analize socialnih omrežij je odkriti osnovne strukture v omrežjih, s čimer se ukvarjamo tudi pri metodi bločnega modeliranja. S to metodo skrčimo skupine enot, ki se nakazujejo znotraj omrežij, in opazujemo, kako so te skupine med seboj povezane (Borgatti in Everett 1992, 91). Enote razvrščamo v skupine na podlagi izbrane enakovrednosti (Wasserman in Faust 1994, 394). Poznamo različne vrste enakovrednosti ter različne pristope k bločnemu modeliranju (Batagelj, Ferligoj in Doreian 1992). V magistrskem delu med seboj ocenjujemo in primerjamo različne metode bločnega modeliranja, in sicer metode neposrednega in posrednega pristopa ter metodi za stohastične modele. Med seboj primerjamo osem različnih metod. Analizo smo izvedli na simuliranih omrežjih s programskim paketom R. Omrežja so bila generirana z različnimi parametri. Simulirali smo 26 tipov omrežij glede na dve velikosti, dve različno veliki skupini, tri različne vrednosti na povezavah in glede na različne bločne modele, vsako simulacijo pa smo ponovili 100-krat. Gre za generirana omrežja, ki so redkejša, konkretneje takšna, kjer so tudi polni bloki redki (delež povezav je tudi tu pod 0.5). Za primerjavo razbitij, ki ga je podala posamezna metoda, s tistim, ki je bilo uporabljeno pri generiranju omrežja, smo uporabili popravljen Randov indeks, mero za podobnosti med razvrstitvama (Rand 1971; Hubert in Arabie 1985, 198). Namen magistrskega dela je oceniti in primerjati različne metode bločnega modeliranja ter ugotoviti na katerih vrstah omrežja se najbolje obnesejo.

Ključne besede: socialno omrežje, bločno modeliranje, simulacija omrežij, strukturna enakovrednost.

Evaluation and comparison of different methods to blockmodeling on simulated networks

Social network analysis is a method that is used in examining relations in sets of units (Hlebec in Kogovšek 2006, 7–9). One of goals of social network analysis is to find the basic structure of the networks, which we are dealing with blockmodeling method. With this method we reduce groups of units into clusters found within the network and observe how these groups are interlinked (Borgatti and Everett 1992, 91). Units are grouped on the basis of the chosen equivalence (Wasserman and Faust 1994, 394). There are different types of equivalence and also different approaches to blockmodeling (Batagelj, Ferligoj in Doreian 1992). In this Master's Thesis we compare different methods of blockmodeling methods, direct and indirect approach and methods for stochastic models. We compare eight different methods. The analysis was performed with simulated networks with the software package R. The networks were generated with different parameters. We simulated 26 types of networks with two different sizes, two different large groups, with three different values of ties and different blockmodels. Each simulation was repeated 100 times. Generated networks are sparse, more precisely they are such that also the complete block are spare (the share of ties is below 0.5). For comparison partition of particular method to that which was used in the generation of networks, we used Adjusted Rands index, measure of the similarity between partitions (Rand 1971; Hubert and Arabia 1985 198). The purpose of this Master's Thesis is to evaluate and compare different methods of blockmodeling and to determine on which types of networks they work best.

Keywords: social network, blockmodeling, simulation networks, structural equivalence.

Kazalo

Kazalo tabel

Kazalo slik

1 Uvod

Analiza socialnih omrežij je metoda, ki se ukvarja s preučevanjem odnosov v množici enot. Socialno omrežje je končna množica sestavljena iz enot (angl. Vertices), med katerimi potekajo relacije oz. povezave (angl. Ties) (Hlebec in Kogovšek 2006, 7–9). Eden izmed ciljev analize socialnih omrežij je odkriti osnovne strukture v omrežjih. Ena izmed takih metod analize socialnih omrežij je bločno modeliranje, s katerim skrčimo skupine enot, ki se nakazujejo znotraj omrežij, in opazujemo, kako so te skupine med seboj povezane (Borgatti in Everett 1992, 91). Cilj bločnega modeliranja je razmejiti osnovno strukturo omrežja s tem, da jo preoblikujemo v manjšo in bolj razumljivo obliko (Batagelj in drugi 2004). Enote razvrščamo v skupine na podlagi izbrane enakovrednosti (Wasserman in Faust 1994, 394). Poznamo različne vrste enakovrednosti: strukturno, regularno, posplošeno in stohastično enakovrednost. Prav tako poznamo različne pristope k bločnemu modeliranju. Tako poznamo posredni in neposredni pristop (Batagelj, Ferligoj in Doreian 1992), ločimo pa tudi med determinističnimi in stohastičnimi pristopi (Nowicki in Snijders 2001).

Cilj magistrskega dela je oceniti in primerjati različne metode bločnega modeliranja. Tema je relevantna, saj evalvacija različnih metod bločnega modeliranja še ni bila narejena predvsem v smislu primerjave med stohastičnimi in determinističnimi metodami, prav tako ni bila narejena za vse vrste omrežij, bločnih modelov in enakovrednosti.

Ugotavljali smo, katere metode se bolje obnesejo na posameznih vrstah omrežij. Ocenjevali smo metode neposrednega in posrednega pristopa ter metodi za stohastične modele. Med seboj smo primerjali osem različnih metod. Analiza je potekala na simuliranih omrežjih. Simulacija in generiranje omrežij sta potekala v statističnem programu R, ki je brezplačni odprtokodni program za statistično analizo podatkov (Blejec 2009, 1). Omrežja smo generirali glede na različne parametre, kot so bločni model, število skupin, velikost skupin in vrednost verjetnosti povezav na polnih blokih. Gre za precej redka generirana omrežja oz. redke polne bloke omrežij, saj so verjetnosti povezav nižje. Izvedli smo 26 simulacij, vsako smo ponovili 100-krat glede na različne parametre, in sicer z izvajanjem osmih različnih metod, iskanjem 100 naključno izbranih razbitij za direktne metode z lokalno optimizacijo ter primerjanjem razbitij posamezne metode z razbitjem, uporabljenem pri generiranem omrežju. Za mero ustreznosti razbitja, ki ga je podala določena metoda, smo vzeli popravljen Randov indeks (ARI, ang.: Adjusted Rand Indeks). Gre za mero podobnosti med razvrstitvama (Rand 1971;

Hubert in Arabie 1985, 198). Na podlagi različnih faktorjev smo primerjali metode med seboj ter ugotavljali, kako se posamezne metode obnesejo glede na posamezen faktor. Z analizo variance smo ugotavljali, kateri učinki in interakcije učinkov najbolj vplivajo na vrednost popravljenega Randovega indeksa.

V magistrskem delu najprej sledi teoretični del, kjer predstavimo analizo socialnih omrežij in definicijo socialnega omrežja. V nadaljevanju se osredotočimo na metodo bločnega modeliranja. Sledi empirični del, kjer so predstavljeni metodologija, generiranje in simuliranje omrežij in na koncu rezultati analize.

2 Analiza socialnih omrežij

2.1 Opredelitev omrežja

Analiza omrežij se ukvarja predvsem z odnosi med enotami, v ospredju njenega preučevanja pa je socialno omrežje (Hlebec in Kogovšek 2006, 7). Na splošno lahko socialno omrežje definiramo kot »celoto, sestavljeno iz niza ali nizov akterjev in relacije ali relacij med njimi« (Wasserman in Faust 1994, 20).

Omrežje je tako sestavljeno iz več delov: enot, vezi in relacij. Enote omrežja so člani ali udeleženci omrežja, lahko gre za osebe, objekte ali dogodke (Hlebec in Kogovšek 2006, 9). Akterji so med sabo povezani z vezmi oz. povezavami, le-te se pojavijo, ko enote med sabo vzpostavijo stik. Relacija oz. relacije pa so povezave med vsemi enotami. Ločimo med simetričnimi in nesimetričnimi relacijami. Ko je odnos med enotami vzajemen ali dvosmeren, gre za simetrične relacije, za nesimetrične pa, ko so odnosi po naravi nevzajemni ali enosmerni (Hlebec in Kogovšek 2006, 9).

2.2 Osnovne značilnosti socialnega omrežja

Poznamo več vrst omrežij in jih delimo glede na različne lastnosti. Omrežja tako lahko razlikujemo glede na velikost ter ločimo med malimi in večjimi omrežji. Med mala omrežja spadajo tista z nekaj deset enotami in povezavami, medtem ko med večja omrežja spadajo tista s po nekaj tisoč enotami in povezavami (de Nooy, Mrvar in Batagelj 2011, 7).

Če želimo omrežja ločiti glede na relacijo, poznamo: neusmerjeno, usmerjeno, splošno in dvovrstno omrežje. Neusmerjeno omrežje sestavljajo simetrične relacije, vse povezave so neusmerjene (angl. Edges). V usmerjenem omrežju so relacije nesimetrične, zato so povezave v omrežju usmerjene (angl. Arcs). Splošno omrežje je, ko imamo v omrežju usmerjene in neusmerjene povezave (de Nooy, Mrvar in Batagelj 2011, 7).

Glede na število množic enot v omrežju ločimo tudi med enovrstnimi in dvovrstnimi omrežji. Enovrstno omrežje je sestavljeno iz množice enot in množice relacij med enotami. Vsaka enota se lahko poveže z drugo enoto. Kot je že zgoraj omenjeno, imamo pri dvovrstnih omrežjih dve množici enot in množico relacij. Relacija povezuje množici enot, medtem ko povezave znotraj vsake množice niso dovoljene (Wasserman in Faust 1994, 35).

Omrežja lahko ločimo tudi glede na uporabljeno mersko lestvico, ki smo jo uporabili za merjenje relacij. Tako lahko omrežja delimo na binarna omrežja, označena omrežja in omrežja z vrednostmi na povezavah. Pri binarnih omrežjih povezava obstaja ali ne obstaja, pri označenih omrežjih ima povezava predznak, pri omrežjih z vrednostmi na povezavah pa so povezave vsaj intervalnega tipa (Žiberna 2007, 5).

Ena izmed delitev, je tudi delitev na popolna in egocentrična omrežja. Pri popolnih omrežjih opazujemo relacije vsake enote z vsemi ostalimi enotami v omrežju. Pri egocentričnih ali osebnih omrežjih pa opazujemo izbrane enote ali ege in njegove povezave do ostalih članov omrežja oz. alterjev (Hlebec in Kogovšek 2006, 11–12).

V magistrskem delu analiza poteka na popolnih enovrstnih binarnih omrežjih.

3 Bločno modeliranje

Med metode analize omrežij spada tudi bločno modeliranje. Glavni namen te metode je reduciranje podatkov, s katerim dobimo poenostavljen model odnosov med oblikami vrst elementov (Borgatti in Everett 1992, 91; Wasserman in Faust 1994; Doreian, Batagelj in Ferligoj 2005). »Z bločnim modeliranjem iščemo skupine akterjev, ki imajo zelo podobne vzorce z drugimi, in interpretiramo vzorec odnosov med skupinami« (Borgatti in Everett 1992, 91). S to metodo seboj podobne enote skrčimo v eno točko modela glede na eno izmed vrst enakovrednosti. Tako dobimo manjše in bolj pregledno omrežje (Mizruchi 1994, 332). Rezultat bločnega modeliranja je torej bločni model, ki je nekakšna manjša prezentacija omrežja s temeljno zgradbo (Doreian, Batagelj in Ferligoj 2005).

Bločni model je sestavljen iz niza pozicij in povezav med pozicijami, ki so opisane z vrsto oz. tipom bloka (Doreian, Batagelj in Ferligoj 2005, 234). Bločni model lahko predstavimo z zmanjšanim grafom ali relacijsko matriko. Točke oz. enote predstavljajo skupine oz. pozicije akterjev, puščice oz. povezave pa predstavljajo odnose med enotami (Doreian, Batagelj in Ferligoj 2005, 169).

Blok lahko definiramo kot »relacijo med dvema skupinama akterjev« (Doreian, Batagelj in Ferligoj 2005, 12). Obstaja več vrst blokov. Tako poznamo diagonalne bloke, s katerimi

preučujemo vezi znotraj skupine. Poznamo pa tudi nediagonalne bloke, ki prikazujejo odnose med skupinami (Doreian, Batagelj in Ferligoj 2005, 12).

3.1 Enakovrednosti

Enote razvrščamo v skupine na podlagi izbrane enakovrednosti. To je matematični pogoj, ki mora biti izpolnjen, da omrežja lahko smatramo za enakovredna (Wasserman in Faust 1994, 394). V dejanskih omrežjih se redko zgodi, da bodo enote med seboj popolnoma enakovredne. Drugi korak v analizi je torej, da določimo, kateremu merilu enakovrednosti v bločnem modelu bomo sledili, s čimer določimo, kaj bomo merili (Wasserman in Faust 1994, 395).

3.1.1 Strukturna enakovrednost

Najpogosteje uporabljena enakovrednost je strukturna enakovrednost. Po Lorrain in White (1971, 63) se definicija strukturne enakovrednosti glasi: »enoti X in Y sta strukturno enakovredni, če je X povezan z vsako enoto iz množice E na enak način kot Y. Ko sta enoti strukturno enakovredni, sta izmenljivi« (Lorrain in White 1971, 63). Matematična definicija strukturne enakovrednosti sledi v nadaljevanju.

Da velja, da sta enoti X in Y strukturno enakovredni, morajo biti izpolnjeni naslednji pogoji (Doreian, Batagelj in Ferligoj 2005, 172):

$s1.\ \ XRY \Leftrightarrow YRX,$

$s2.\ \ XRX \Leftrightarrow YRY,$

$s3.\ \ \forall Z \ni E\backslash\{X,Y\}: (XRZ \Leftrightarrow YRZ),$

$s4.\ \ \forall Z \ni E\backslash\{X,Y\}: (ZRX \Leftrightarrow ZRY).$

Po Doreian, Batagelj in Ferligoj (2005, 173) naj bi za to vrsto enakovrednosti obstajale štiri vrste idealnih blokov, ki so prikazani na Sliki 3.1. V prikazanih primerih blokov spodnja dva bloka veljata samo za diagonalne, medtem ko zgornja dva primera veljata za vse bloke (glej Slika 3.1).

Slika 3.1: Primeri idealnih blokov za strukturno enakovrednost

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

3.1.2 Regularna enakovrednost

White in Reitz (1983, 200) regularno enakovrednost definirata tako: »Dve enoti sta regularno enakovredni, če sta enako povezani s skupinami enakovrednih enot.« Regularna enakovrednost je posplošitev strukturne, zato je vsaka strukturna enakovrednost tudi regularna (Doreian, Batagelj in Ferligoj 2005,174).

Matematična definicija regularne enakovrednosti se tako glasi (White in Reitz 1983, 200): Enakovrednost ≡ na P je regularna enakovrednost na omrežju $G = (P, R)$ če in samo če za vsak par $a, b, c \in P$ $a \equiv b$ pomeni da:

1. za vsak aRc obstaja $d \in P$ kjer je bRd in $d \equiv c$,
2. za vsak cRa obstaja $d \in P$ kjer je dRb in $d \equiv c$.

Regularni enakovrednosti naj bi ustrezali dve vrsti idealnih blokov: prazen blok z vsemi vrednostmi 0 in regularni blok, ki ima v vsaki vrstici in v vsakem stolpcu vsaj eno 1 (Ferligoj in Batagelj 1996, 166; Doreian, Batagelj in Ferligoj 2005, 174). Primera idealnih blokov za regularno enakovrednost sta (Doreian, Batagelj in Ferligoj 2005, 174):

Slika 3.2: Primera idealnih blokov za regularno enakovrednost

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

0	1	0	0	0
1	0	1	1	0
0	1	0	1	0
1	1	0	0	0

3.1.3 Posplošena enakovrednost

Pojem posplošene enakovrednosti so predstavili Doreian, Batagelj in Ferligoj (1994). Pri tej vrsti enakovrednosti dopustimo, da lahko vsak blok glede na dano razvrstitev sledi različni enakovrednosti. Vsak blok v matriki pa ima lahko tudi poseben vzorec, kjer so lahko določene vrste enakovrednosti posebni primeri bolj posplošenih vzorcev (Doreian, Batagelj in Ferligoj 1994, 2).

3.1.4 Stohastična enakovrednost

Poznamo tudi stohastično enakovrednost, pri kateri gre za posplošitev strukturne enakovrednosti. Strukturna enakovrednost obsega stohastično enakovrednost, medtem ko obratno ne velja. »Dva akterja sta stohastično enakovredna, če imata enako verjetnostno porazdelitev povezav do drugih enot« (Anderson, Wasserman in Faust 1992, 140).

Definicija stohastične enakovrednosti (Anderson, Wasserman in Faust 1992, 140):
»Glede na dan stohastični graf, ki ga predstavlja niz naključnih matrik X, sta akterja i in i' stohastično enakovredna, če in samo če verjetnost vsakega dogodka, opisanega z matriko X, ostane nespremenjena z izmenjavo akterjev i in i'.«

3.2 Bločni modeli

3.2.1 Različni pristopi

Poznamo posredni in neposredni pristop k bločnemu modeliranju (Batagelj, Ferligoj in Doreian 1992). Pri posrednem pristopu je najpomembnejša zahteva, da je izbrana različnost usklajena z izbrano enakovrednostjo (Batagelj, Doreian in Ferligoj 1992, 122). Pri neposrednem pristopu pa neposredno iščemo razbitje, ki najbolj ustreza izbrani enakovrednosti, ustreznost razbitij pa se meri s kriterijsko funkcijo (Batagelj, Ferligoj in Doreian 1992, 66).

Ločimo pa tudi med deterministčnimi in stohastičnimi pristopi. Razlika med tema dvema pristopoma je, da stohastični pristopi temeljijo na verjetnostnem modelu, medtem ko to za deterministične pristope ne velja (Nowicki in Snijders 2001).

3.2.1.1 Posredni pristop

Pri posrednem pristopu najprej določimo različnost med pari enot, na podlagi tega pa izberemo ustrezno enakovrednost (Doreian, Batagelj in Ferligoj 2005, 177). Mera različnosti d je usklajena z izbrano enakovredostjo, če za vsak par enot velja $X_i \sim X_j \leftrightarrow d(X_i, X_j) = 0$ (Doreian, Batagelj in Ferligoj 2005, 182).

Obstaja več mer različnosti, zelo malo pa jih je usklajeno s strukturno enakovrednostjo. Kot mero d tako lahko uporabimo popravljeno ali prilagojeno evklidsko razdaljo, ki je najbolj usklajena s to enakovrednostjo (Burt in Minor 1983 v Batagelj, Ferligoj in Doreian 1992, 71; Doreian, Batagelj in Ferligoj 2005, 181):

$$d(X_1, X_2) = \sqrt{(r_{ii} - r_{jj})^2 + (r_{ij} - r_{ji})^2 + \sum_{\substack{s=1 \\ s \neq i,j}}^{n} \left(\left(r_{is} - r_{js}\right)^2 + \left(r_{si} - r_{sj}\right)^2 \right)}.$$

Pri posrednem pristopu k bločnemu modeliranju gre za hierarhično združevanje v skupine. Ta temelji na zaporednem združevanju dveh ali več skupin v novo skupino (Ferligoj 1989, 61). Potek združevanja grafično ponazorimo z drevesom združevanja, s t. i. dendrogramom. Pri dendrogramu so listi drevesa enote, točke združitve pa skupine (Ferligoj 1989, 68).

Mere različnosti d med novo skupino in skupino C_k lahko določimo na različne načine: (Doreian, Batagelj in Ferligoj 2005, 148–149; Ferligoj 1989, 64):

- minimalna metoda ali enojna povezanost (Florek in drugi 1951; Sneath 1957):

$$d(C_i \cup C_j, C_k) = \min(C_i, C_k), d(C_j, C_k)),$$

- maksimalna metoda ali polna povezanost (McQuitty 1960):

$$d(C_i \cup C_j, C_k) = \max(C_i, C_k), d(C_j, C_k)),$$

- McQuittyjeva (McQuitty 1966, 1967):

$$d(C_i \cup C_j, C_k) = \frac{\mathrm{d}(C_i,C_k) + d(C_j,C_k)}{2}.$$

Različnosti lahko določamo tudi na bolj zapletene načine, tako da upoštevamo sestavo posameznih skupin. Eden izmed takih načinov je tudi (Doreian, Batagelj in Ferligoj 2005, 148–149; Ferligoj 1989, 64):

- Wardova metoda – povzemamo formulo, ki temelji na Lance-Williamsovem obrazcu (Ferligoj 1989, 70–72; Žiberna 2007, 27):

$$d(C_i \cup C_j, C_k) = \frac{(n_i+n_k)}{(n_i+n_j+n_k)}\mathrm{d}(C_k, C_i) + \frac{(n_j+n_k)}{(n_i+n_j+n_k)}\mathrm{d}(C_k, C_j) - \frac{n_k}{(n_i+n_j+n_k)}\mathrm{d}(C_i, C_j).$$

3.2.1.2 Neposredni pristop

Drugi pristop k bločnemu modeliranju je neposredni pristop. Pri tem pristopu neposredno sestavimo kriterijsko funkcijo in potem uporabimo algoritem za lokalno optimizacijo. Kriterijska funkcija mora meriti odstopanje dejanskih blokov od pripadajočih idealnih blokov glede na model, hkrati pa mora biti občutljiva za enakovrednost, ki jo obravnavamo (Doreian, Batagelj in Ferligoj 2005, 185–186).

V primeru, da izhajamo iz razvrstitve $C = \{C_1, C_2, \ldots . C_k\}$, množico vseh idealnih blokov označimo z $\mathcal{B}(C_u, C_v)$ glede na blok $R(C_u, C_v)$, lahko celotno napako razvrstitve C izrazimo s kriterijsko funkcijo (Doreian, Batagelj in Ferligoj 2005, 186):

$$P(C) = \sum_{C_u, C_v \in C} \min_{B \in \mathcal{B}(C_u, C_v)} \delta(R(C_u, C_v), B).$$

Člen $\delta(R(C_u, C_v), B)$ v tem primeru meri razliko oz. napako med blokom $(R(C_u, C_v), B)$ in idealnim blokom B, funkcija δ pa mora biti usklajena z izbranim tipom enakovrednosti (Doreian, Batagelj in Ferligoj 2005, 186).

Neposredne metode kriterijsko funkcijo že vključijo v postopek iskanja optimalnega razbitja. Tako smo lahko prepričani, da bomo našli vsaj lokalno optimalno razbitje ali razbijta, česar nam posredne metode ne zagotavljajo. Prednost neposrednih metod je tudi v zmožnosti njihovega prilagajanja (Žiberna 2007, 43). Pravzaprav gre za metode posplošenega bločnega modeliranja, ki pri optimizaciji uporabljajo lokalno optimizacijo. Lokalna optimizacija je postopek, s katerim iščemo čim boljšo rešitev, ponavljamo pa ga toliko časa, da dane razvrstitve ne moremo več izboljšati. Postopek ne zagotavlja, da dobimo najboljšo rešitev, se pa lahko najboljši rešitvi približamo, če postopek ponovimo z več različnimi začetnimi razvrstitvami (Ferligoj 1989, 87–88).

3.2.1.3 Stohastično bločno modeliranje

Fienberg in Wasserman (1981) ter Holland, Laskey in Leinhardt (1983) so razširili koncept bločnega modeliranja na stohastično različico. Tako poznamo tudi t. i. metode za stohastične bločne modele oz. t. i. »mixture models« (modele na podlagi mešanic) (Daudin, Picard in Robin 2008). Stohastični bločni model lahko definiramo kot verjetnostno porazdelitev za grafe, katerih akterji v omrežju so razdeljeni v podskupine, ki jih imenujemo bloki (Snijders in Nowicki 1997, 77; Holland, Laskey in Leinhardt 1983). Stohastični model je t. i. »mixture model« model, ki opisuje, kako povezave povezujejo točke, upošteva pa tudi heterogenost med točkami (Daudin, Picard in Robin 2008, 175). Pri tej metodi je verjetnost za povezavo pri binarnih omrežjih pri omrežjih z vrednostmi na povezavah odvisna le od tega, katerima skupinama pripadata začetna in končna enota povezave (Žiberna 2007, 258).

Trenutni postopki ocenjevanja parametrov naključnih grafov v t. i. »mixture models« temeljijo na približevanju verjetnosti. Temelj takih strategij sta EM-algoritem (expectation-maximization) (Dempster, Laird in Rubin 1977) in bayesianski pristop (Ambroise in Matias

2012, 4). Zlasti v primerih t. i. »mixture models« natančen izračun te pogojne distribucije ni mogoč. Zaradi tega so približni izračuni narejeni s t. i. »variacijskimi« EM-algoritmi in bayesianskimi strategijami (Daudin, Picard in Robin 2008; Latouche, Birmelé in Ambroise 2012; Picard in drugi 2009; Zanghi, Ambroise in Miele 2008).

4 CILJI IN RAZISKOVALNA VPRAŠANJA

Glavni namen magistrskega dela je bilo oceniti in primerjati različne metode za bločno modeliranje. Ocenili smo primernost posamezne metode za posamezne vrste omrežij, želeli pa smo tudi oceniti, katera metoda oz. metode se najboljše obnese na posamezni vrsti omrežja oz. bločnega modela.

4.1 METODOLOGIJA

Pristopi k bločnemu modeliranju so implementirani v različnih programih. Posredni in neposredni pristop sta implementirana v programu za analizo omrežij Pajek (Batagelj in Mrvar 2011) in v statističnem programu R v paketu blockmodeling (Žiberna 2010). Stohastični pristopi so implementirani v R-paketku mixer (Ambroise in drugi 2013).

Ocenjevanje različnih metod bločnega modeliranja je potekalo v statističnem programu R. R je celovita računalniška platforma in jezik za statistično analizo podatkov, ki podpira širok nabor podatkovnih struktur, ima odlično grafično zmogljivost in zbirko statističnih metod. Temelji na računalniškem jeziku S, ki je učinkovit jezik in okolje za statistično analizo podatkov (Blejec 2009, 1). Naša analiza je potekala s pomočjo R-paketa blockmodeling (Žiberna 2010) ter paketka mixer (Ambroise in drugi 2013). R-paket blockmodeling je prvotno namenjen implementaciji posplošenega bločnega modeliranja, so pa v njem tudi funkcije za računanje strukturne in regularne enakovrednosti (Žiberna 2010).

V delu smo uporabili programsko verzijo R 3.0.0 in operacijski sistem Windows 7.

Analiza je potekala na simuliranih omrežjih. Omrežja so bila generirana glede na različne parametre, kot so razbitja, bločni modeli, enakovrednosti, velikosti omrežja (Žiberna 2009, 106). Pri analizi smo uporabili tudi R-paket e1071. Iz tega paketa smo uporabili funkcijo classAgreement, ki računa popravljen Randov indeks (Meyer in drugi 2014). S popravljenim Randovim indeksom, ki je mera podobnosti med razvrstitvama (Rand 1971; Hubert in Arabie

1985, 198), smo primerjali razbitja, ki ga je podala posamezna metoda s pravim razbitjem, torej tistim, ki bo uporabljeno pri generiranju omrežja. Popravljen Randov indeks smo vzeli kot mero ustreznosti oz. kvalitete razbijta, ki ga poda določena metoda in s katerim pravzaprav poda kvaliteto metode. Ocenjevali smo, kako se metode obnesejo na različnih vrstah omrežij. Metode smo primerjali med sabo in skušali ugotoviti, katera metoda oz. metode so najboljše za posamezne vrste omrežij.

4.1.1 Ocenjevane metode

Ocenjevali in primerjali smo metode neposrednega in posrednega pristopa ter metodi za stohastične modele. Posamezne metode so opisane v nadaljevanju.

4.1.1.1 Metode neposrednega pristopa

Pri neposrednem pristopu neposredno sestavimo kriterijsko in nato uporabimo algoritem za lokalno optimizacijo (Doreian, Batagelj in Ferligoj 2005, 185–186). Pri metodah neposrednega pristopa smo uporabili metode binarnega bločnega modeliranja, ki smo jih poimenovali *binStr, binStrU, binReg*, ter metodi homogenega bločnega modeliranja (*hom, homOm*).

Metode binarnega bločnega modeliranja

Binarno bločno modeliranje je namenjeno analizi na binarnih omrežjih. Pojem binarnega bločnega modeliranja je temeljito predstavljen v Doreian in drugi (2005), kjer se nanaša na posplošeno bločno modeliranje na binarnih omrežjih (Doreian in drugi 2005). Omejeno je na binarna omrežja, pri katerih povezava med dvema akterjema obstaja ali ne obstaja. Ta metoda torej ne upošteva vrednosti vezi, obravnava jih kot obstoječe ali kot neobstoječe (Doreian in drugi 2005; Žiberna 2007, 5). Neskladnost oz. napako bloka pri tej metodi načeloma merimo s številom prisotnosti oz. odsotnosti povezav tam kjer niso oz. so predvidene (Žiberna 2009, 102). Ocenjujemo tri različne metode binarnega bločnega modeliranja.

- *binStr* – pristop, kjer dovoljujemo bloke strukturne enakovrednosti (null in complete). Dovoljujemo torej prazne in polne bloke. Primeri idealnih blokov za strukturno enakovrednost so predstavljeni na Sliki 3.1. Pri tem pristopu kot napako smatramo povezave v praznih blokih in odsotnost povezav v polnih blokih. Ta metoda je primerna, kadar želimo bloke, ki imajo gostoto večjo kot 0.5, kategorizirati kot polne, tiste z manjšo gostoto kot 0.5 pa kategorizirati kot prazne.

- *binStrU* – pristop, kjer dovoljujemo bloke strukturne enakovrednosti (null in complete) z različno uteženimi bloki. Pri tem pristopu se uporabi strukturna enakovrednost, kjer so polni in prazni bloki drugače uteženi. Če želimo bloke z gostoto, večjo kot d, kategorizirati kot polne bloke in z gostoto, manjšo ali enako d, pa kot prazne bloke, je primerna utež za prazne bloke 1 in za polne bloke $d\ /(1-d)$ ali $1-d$ za prazne bloke in d za polne bloke, saj je pomembno samo razmerje med utežmi. Prednost takšnega pristopa je, da so lahko polni bloki gosti, kolikor se da, čeprav še vedno ni zahteve po enakosti gostot med vrsticami in med stolpci (Žiberna 2013, 103).

- *binReg* – pristop, kjer dovoljujemo bloke regularne enakovrednosti (null, complete, regular). Poleg praznih in polnih blokov dovoljujemo tudi regularne bloke, ki imajo v vsaki vrstici in v vsakem stolpcu vsaj eno 1 (Ferligoj in Batagelj 1996, 166; Doreian, Batagelj in Ferligoj 2005, 174). Primera idealnih blokov za regularno enakovrednost sta predstavljena na Sliki 3.2.

Metodi homogenega bločnega modeliranja

Glavna ideja homogenega bločnega modeliranja je, da išče taka razbitja, kjer je vsota mere variabilnosti znotraj blokov glede na ostale bloke najnižja. Mera variabilnosti znotraj bloka predstavlja neskladnost bloka. Uporabljata se dve meri variabilnosti: vsota kvadratnih odklonov in vsota absolutih odklonov (Žiberna 2007, 61). V našem delu ocenjujemo metodo homogenega bločnega modeliranja, kjer je mera variabilnosti vsota kvadratov. Omejitev pristopa homogenega bločnega modeliranja, pri katerem je mera variabilnosti vsota kvadratov, je v tem, da je nični blok poseben primer vseh tipov blokov, ker ta pristop zahteva, da so določene vrednosti celic ali funkcij homogene in imajo enakovredno vrednost. Na splošno ni definirano, s katero vrednostjo morajo biti enakovredne, ta vrednost je lahko tudi 0. V takih primerih se zaradi idealne strukture vsi drugi tipi blokov zreducirajo na tip ničnega bloka. Če se uporabi pristop vsote kvadratov, je optimalna vrednost za popoln blok povprečje vseh vrednosti in bo nični blok imel večje neskladje kot cel blok (Žiberna 2013, 104). V našem magistrskem delu ocenjujemo dve metodi homogenega bločnega modeliranja.

- *hom* – pristop brez omejitev, pri katerem je mera variabilnosti vsota kvadratov (Žiberna 2007).

- *homOm* – pristop z omejenimi popolnimi bloki (null in complete) (Žiberna 2007). Žiberna (2013) predlaga, da se problem t. i. »praznega bloka« rešuje tako, da se omeji vse bloke, ki niso prazni. Omejimo jih tako, da je vrednost, od katere se računajo odkloni, večja ali enaka neki vnaprej določeni vrednosti, do katere bi vrednosti morale biti homogene. Če uporabimo samo prazne in polne bloke znotraj pristopa vsote kvadratov, želimo, da je povprečje bloka enako 0 ali blizu 0 ali višje ali enako vnaprej določeni vrednosti. Za polne bloke je tako razumna odločitev, da je ta vrednost dvakrat višja kot povprečje omrežja. V primeru, ko so dovoljeni samo prazni in polni bloki, se le-te klasificira kot polne, če je njihovo povprečje večje kot povprečje celotnega omrežja (Žiberna 2013, 104–105).

Metoda posrednega pristopa

Pri posrednem pristopu najprej določimo različnost med pari enot, ter na podlagi tega izberemo enakovrednost, ki bi bila ustrezna (Doreian, Batagelj in Ferligoj 2005, 177). Ocenjujemo eno metodo posrednega pristopa.

- *sedist* – računa razdaljo glede na strukturno enakovrednost (Lorrain in White 1971) in jo računa med enotami enovrstnega omrežja (Žiberna 2007). Za določanje mere različnosti uporabljamo Wardovo metodo, ki jo izračunamo na podlagi popravljene ali prilagojene evklidske razdalje, saj je ta najbolj usklajena s strukturno enakovrednostjo (Burt in Minor 1983 v Batagelj, Ferligoj in Doreian 1992, 71; Doreian, Batagelj in Ferligoj 2005, 181).

Metodi za stohastične modele

Koncept bločnega modeliranja je bil razširjen tudi na stohastično različico (Fienberg in Wasserman 1981; Holland, Laskey in Leinhardt 1983). Osnovna lastnost metod stohastičnega bločnega modeliranja je, da je verjetnost povezave odvisna samo od pripadnosti skupin (Žiberna 2007, 158). Ocenjujemo dve metodi za stohastične modele: variacijsko in bayesiansko metodo. Ocenjevanje različnih metod bločnega modeliranja še ni bilo narejeno v smislu primerjave med determinističnimi in stohastičnimi metodami. Eden izmed kriterijev, da smo izbrali ravno ti dve metodi pa je tudi, da sta obe implementirani v programu R, in sicer v paketu mixer (Ambroise in drugi 2013).

- *mixVar* – gre za variacijsko metodo, kjer se uporablja ICL-kriterij, ki ustreza asimptotičnemu približevanju za integrirano klasifikacijsko verjetnost (Integrated Classification Likelihood) (Ambroise in drugi 2013). Ta metoda se nanaša na metodo, opisano v Daudin, Picard in Robin (2008).

- *mixBay* – pri bayesianski metodi se uporablja integrirana bayesianska variacijska verjetnost (Integrated Likelihood variational Bayes), ki temelji na variacijskem (neasimptotičnem) približevanju integrirani opazovani verjetnosti (Integrated observed Likelihood) (Ambroise in drugi 2013). Bayesianska metoda se nanaša na metodo, opisano v Latouche in drugi (2008)

4.2 Popravljen Randov indeks

Da ocenimo kvaliteto razbijta, ki ga poda določena metoda, moramo izračunati, kako se dve razvrstitvi ujemata. Ena najbolj razširjenih mer podobnosti med dvema razvrstitvama je Randov indeks (RI, ang: Rand Indeks), ki ga lahko predstavimo z enačbo (Rand 1971):

$$RI = \frac{a+d}{a+b+c+d} = \frac{a+d}{\frac{n}{2}}$$

Ker ima ta mera med drugim tudi to pomanjkljivost, da njena srednja vrednost dveh naključnih razvrstitev ne zavzame konstantne vrednosti (Yeung in Ruzzo 2001, 763), sta Hubert in Arabie (1985) predlagala izboljšavo.

V našem delu bomo tako uporabili popravljen Randov indeks (ARI, ang.: Adjusted Rand Indeks), (Rand 1971; Hubert in Arabie 1985, 198). Z njim bomo primerjali razbitja, ki ga bo podala posamezna metoda s pravim razbitjem, torej tistim, ki bo uporabljeno pri generiranju omrežja. Pri meri popravljenega Randovega indeksa vrednost 1 pomeni popolno ujemanje dveh razvrstitev, 0 pomeni takšno ujemanje, kot če bi bile skupine (takšnih velikosti) generirane slučajno (Rand 1971; Hubert in Arabie 1985, 198).

Tabela 4.1: Kontingenčna tabela za izračun popravljenega Randovega indeksa

U/V	V_1	V_2	…	V_c	$Vsota$
U_1	n_{11}	n_{11}	…	n_{1c}	$n_{1.}$
U_2	n_{21}	n_{22}	…	n_{2c}	$n_{2.}$
⋮	⋮	⋮	…	⋮	⋮
U_R	n_{R1}	n_{R2}	…	n_{4C}	$n_{R.}$
$Vsota$	$n_{.1}$	$n_{.2}$	…	$n_{.3}$	$n \ldots = n$

Enačbo za popravljen Randov indeks lahko zapišemo kot (Hubert in Arabie 1985, 198) :

$$\frac{\text{Randov indeks} - \text{pričakovani Randov indeks}}{\text{maksimalni Randov indeks} - \text{pričakovani Randov indeks}} =$$

$$\frac{\binom{n}{2}(a+d)-((a+b)(a+c)+(c+d)(b+d))}{\binom{n}{2}^2-((a+b)(a+c)+(c+d)(b+d))}.$$

Zapis za popravljen Randov indeks iz Tabele 4.1 lahko zapišemo kot enačbo (Hubert in Arabie 1985, 198):

$$ARI = \frac{\binom{n}{2}\sum_{i=1}^{R}\sum_{j=1}^{C}\binom{n_{ij}}{2} - \sum_{i=1}^{R}\binom{n_{i\cdot}}{2}\sum_{i=1}^{C}\binom{n_{\cdot j}}{2}}{\frac{1}{2}\binom{n}{2}\left(\sum_{i=1}^{R}\binom{n_{i\cdot}}{2} + \sum_{i=1}^{C}\binom{n_{\cdot j}}{2} - \sum_{i=1}^{R}\binom{n_{i\cdot}}{2}\sum_{i=1}^{C}\binom{n_{\cdot j}}{2}\right)}.$$

Popravljen Randov indeks bomo vzeli kot mero ustreznosti oz. kvalitete razbijta, ki ga poda določena metoda in s katerim pravzaprav poda kvaliteto metode.

4.3 Simuliranje podatkov

Posamezna simulacija je potekala s ponavljanjem naslednjih postopkov:

- število ponovitev: 100,
- generiranje omrežij glede na različne parametre,
- izvajanje različnih metod: *binStr*, *binStrU*, *binReg*, *hom*, *homOm*, *sedist*, *mixVar*, *mixBay*,
- iskanje 100 naključno izbranih razbitij za direktne metode z lokalno optimizacijo,
- primerjanje razbitij posamezne metode z razbitjem uporabljenim pri generiranem omrežju s popravljenim Randovim indeksom.

4.3.1 Generiranje omrežij

Pri generiranju omrežij smo izhajali iz različnih bločnih modelov in verjetnosti po povezavah za polne bloke.

Omrežja smo generirali glede na različne vhodne parametre:

- bločni model,
- verjetnosti določene po blokih,
- število enot (povsod 60),
- omrežja iz različnih razbitij:
    - omrežja, sestavljena iz dveh razbitij: c(30, 30) in c(40, 20),
    - omrežja, sestavljena iz treh razbitij: c(20, 20, 20) in c(40, 10, 10).

Slučajne vrednosti so bile generirane iz funkcije *rbinom*, ki generira slučajne vrednosti iz binomske porazdelitve, verjetnosti pa so bile določene po blokih.

Izhajali smo iz različnih bločnih modelov, predstavljenih z bločnimi matrikami, ki temeljijo na strukturni enakovrednosti. Polne in prazne bloke smo pridobili z različnimi vrednostmi verjetnosti po povezavah. Za polne bloke smo uporabili vrednosti 0.5 in 0.3, za prazne bloke pa samo vrednost 0.1. Te vrednosti so predstavljene v Tabeli 4.2. Omrežja so sestavljena iz različnih razbitij oz. različnega števila skupin. Pri prvih petih modelih imamo po dva načina generiranja omrežij z enakim številom enot v razbitju in po dva načina generiranja omrežij z različnim številom enot v razbitju. Lastnosti po posameznem načinu generiranja omrežij so predstavljene v Tabeli 4.2.

Uporabljeni bločni modeli so v nadaljevanju predstavljeni z bločnimi matrikami (BM). Podani so tudi primeri generiranih omrežij, ki izhajajo iz predstavljenih bločnih matrik.

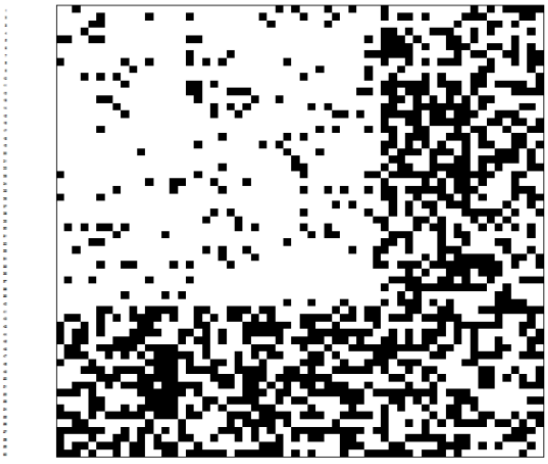
$$BM_1 = \begin{bmatrix} \text{null} & \text{com} \\ \text{com} & \text{com} \end{bmatrix}$$

Bločni model s tremi polnimi (com) in z enim praznim blokom (null). Iz te bločne matrike smo izhajali pri načinu generiranja omrežij *1*, *2*, *7* in *8*. Gre za model center-periferija s centralno skupino, znotraj katere so skupine povezane med sabo in z enotami iz drugih skupin. Enote iz drugih (perifernih) skupin so povezane s centralno skupino, vendar niso povezane med sabo (Batagelj in drugi 2004, 236).

Slika 4.1: Primer generiranja omrežja *1*

Slika 4.2: Primer generiranja omrežja *2*

Slika 4.3: Primer generiranja omrežja *7*



Slika 4.4: Primer generiranja omrežja *8*

Na zgornjih slikah so prikazani primeri generiranih omrežij z bločno matriko BM_1. V prvih dveh primerih (Slika 4.1 in Slika 4.2) gre za enako število enot v posameznem razbitju, omrežji pa se razlikujeta po vrednosti verjetnosti v polnem bloku. V drugih dveh primerih gre za omrežji z različnim številom enot v posameznem razbitju. Omrežji se prav tako razlikujeta po tem, da je na Sliki 4.3 vrednost polnega bloka 0.5, na Sliki 4.4 pa je vrednost polnega bloka 0.3. V obeh primerih ima edini prazni blok tudi največje število enot.

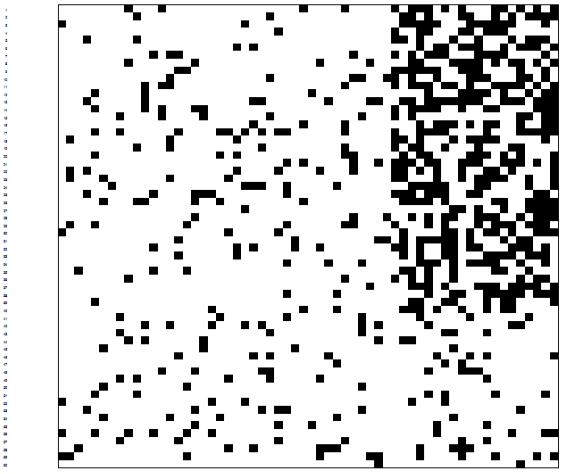
$$BM_2 = \begin{bmatrix} \text{null} & \text{com} \\ \text{null} & \text{null} \end{bmatrix}$$

Iz bločnega modela s tremi praznimi (null) in enim polnim blokom (com) izhajamo pri načinu generiranja omrežij *3*, *4*, *9* in *10*.

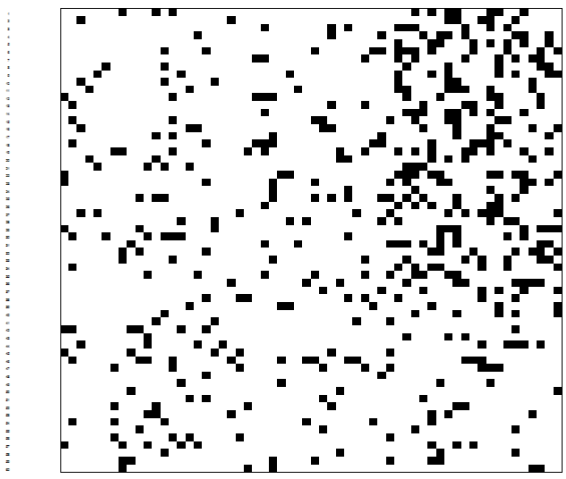
Slika 4.5: Primer generiranja omrežja *3*

Slika 4.6: Primer generiranja omrežja *4*

Slika 4.7: Primer generiranja omrežja *9*



Slika 4.8: Primer generiranja omrežja *10*



Primeri generiranj omrežij, pri katerih izhajamo iz matrike BM_2, so prikazani na zgornjih slikah. Iz Slike 4.5 in Slike 4.6 je razvidno, da gre za omrežji z enakim številom enot v posameznem razbitju, pri drugih dveh omrežjih (Slika 4.7 in Slika 4.8) pa gre za različno število enot v posameznem omrežju. V primeru generiranj omrežij *3* in *9* (Slika 4.5 in Slika 4.7) gre za vrednost verjetnosti polnega bloka 0.5 pri ostalih dveh primerih pa za vrednost verjetnosti polnega bloka 0.3.

$$BM_3 = \begin{bmatrix} \text{com} & \text{null} \\ \text{null} & \text{com} \end{bmatrix}$$

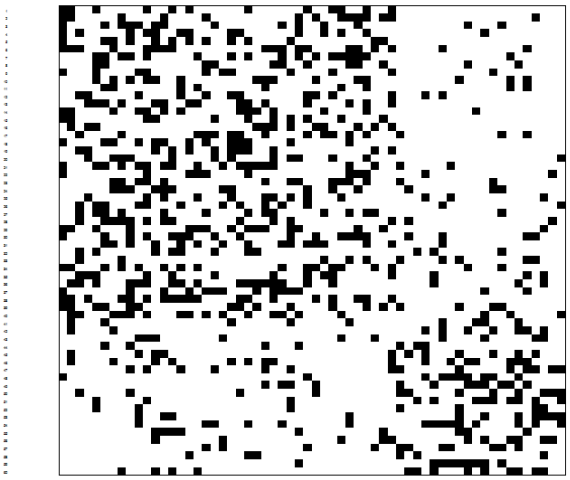
Bločni model BM_3 je predstavljen z bločno matriko, ki ima dva polna bloka (com) po diagonali in dva prazna bloka (null) izven nje. Gre za popolnoma simetrično matriko, iz katere smo izhajali pri načinu generiranja omrežij *5*, *6*, *11* in *12*. Gre za bločni model kohezivnih skupin, kjer so povezave med enotami znotraj vsake skupine in brez povezav med skupinami (Doreian, Batagelj in Ferligoj 2005, 236).

Slika 4.9: Primer generiranja omrežja *5*

Slika 4.10: Primer generiranja omrežja *6*

Slika 4.11 : Primer generiranja omrežja *11*

Slika 4.12: Primer generiranja omrežja *12*



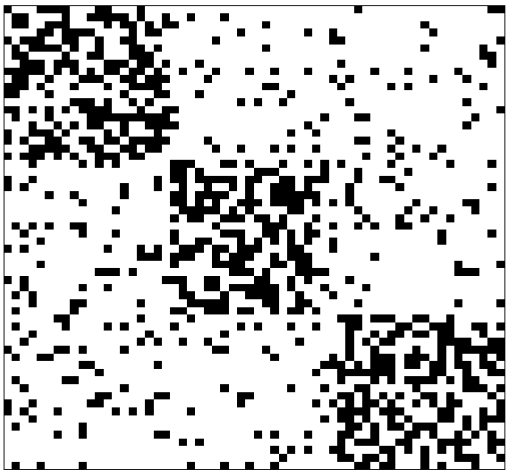
Prikazani so primeri generiranja omrežja z bločno matriko BM_3. Kot že pri ostalih primerih se omrežja med sabo razlikujejo po vrednosti verjetnosti polnega bloka in po različnem številu enot v posameznem razbitju. Na Sliki 4.11 in Sliki 4.12 sta prikazani omrežji, ki imata največ

enot v enem izmed polnih blokov.

$$BM_4 = \begin{bmatrix} \text{com} & \text{null} & \text{null} \\ \text{null} & \text{com} & \text{null} \\ \text{null} & \text{null} & \text{com} \end{bmatrix}$$

Pri načinu generiranja omrežij *15, 16, 19* in *20*, ki so bila sestavljena iz treh razbitij, smo uporabili popolnoma simetrično matriko, ki ima tri polne bloke (com) po diagonali in šest praznih blokov izven nje. Tudi v tem primeru gre za model kohezivnih skupin.

Slika 4.13: Primer generiranja omrežja *15*



Slika 4.14: Primer generiranja omrežja *16*

Slika 4.15: Primer generiranja omrežja *19*

Slika 4.16: Primer generiranja omrežja *20*

V zgornjih primerih so prikazana omrežja, ki temeljijo na modelu kohezivnih skupin (BM_4) s tremi razbitji. Na Sliki 4.13 in Sliki 4.14 imamo primera z enakim številom enot v

posameznem razbitju, razlikujeta se po vrednosti verjetnosti polnega bloka. Na Sliki 4.15 in Sliki 4.16 imamo primera z različnim številom enot v posameznem razbitju. Omrežji imata en večji in dva manjša polna bloka. Prav tako se razlikujeta po verjetnosti polnega bloka, kjer ima primer generiranja omrežja *20* manjšo vrednost verjetnosti (0.3).

$$BM_5 = \begin{bmatrix} \text{com} & \text{com} & \text{com} \\ \text{com} & \text{com} & \text{null} \\ \text{com} & \text{null} & \text{null} \end{bmatrix}$$

Pri načinu generiranja omrežij *17, 18, 21* in *22* smo izhajali iz bločne matrike s petimi polnimi (com) in tremi praznimi (null) bloki. Gre za bločni model center-periferija.

Slika 4.17: Primer generiranja omrežja *17*

Slika 4.18: Primer generiranja omrežja *18*

Slika 4.19: Primer generiranja omrežja *21*

Slika 4.20: Primer generiranja omrežja *22*

Kot pri ostalih bločnih modelih smo tudi tukaj prikazali štiri primere generiranj omrežij, ki izhajajo iz bločnega modela center-periferija. Pri vseh primerih gre za omrežja s po tremi razbitji, v prvih dveh primerih (Slika 4.17 in Slika 4.18) gre za enako število enot, v drugih dveh primerih (Slika 4.19 in Slika 4.20) pa za različno število enot v posameznem razbitju. Na Sliki 4.19 in Sliki 4.20 je še razvidno, da je v praznih blokih manj enot kot v polnih blokih.

Pri načinu generiranja omrežij *13*, *14*, *23*, *24*, *25* in *26* izhajamo iz vrednosti po povezavah na polnih bloki. Za polne bloke smo na posameznih omrežjih uporabili po dve vrednosti (0.5 in 0.3). Tudi v tem primeru smo uporabili dve različno veliki omrežji z različnim številom enot v razbitju oz. različno velike skupine.

$$BM_6 = \begin{bmatrix} \text{com} & \text{com} \\ \text{com} & \text{null} \end{bmatrix}$$

Pri načinu generiranja omrežij *13* in *14* izhajamo iz bločnega modela center-periferija, kjer je bločna matrika sestavljena iz treh polnih in enega praznega bloka.

Slika 4.21: Primer generiranja omrežja *13*

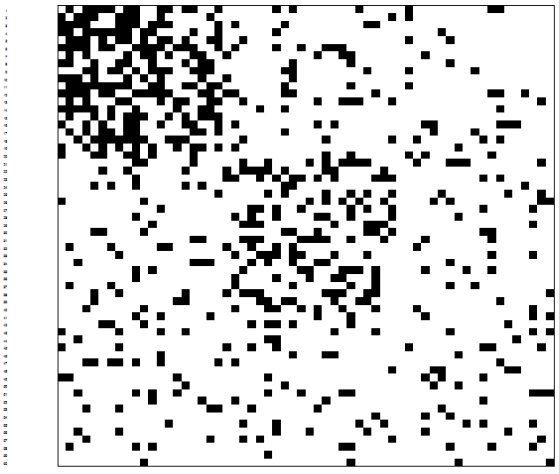
Slika 4.22: Primer generiranja omrežja *14*

Prikazana sta primera generiranja omrežja, kjer izhajamo iz bločne matrike BM_6. Kot že omenjeno, sta pri polnih blokih uporabljeni obe vrednosti za polne bloke. Pri Sliki 4.21 gre za omrežje, sestavljeno iz enakega števila enot v posameznem razbitju, pri Sliki 4.22 pa gre za različno število enot v posameznem razbitju. V drugem primeru (Slika 4.22) ima blok z največjim številom enot v razbitju tudi večjo vrednost verjetnosti polnega bloka (0.5).

$$BM_7 = \begin{bmatrix} \text{com} & \text{null} & \text{null} \\ \text{null} & \text{com} & \text{null} \\ \text{null} & \text{null} & \text{null} \end{bmatrix}$$

Pri načinu generiranja omrežij *23* in *24* imamo tri razbitja ter bločni model iz dveh polnih blokov. Gre za kohezivni bločni model.

Slika 4.23: Primer generiranja omrežja *23*



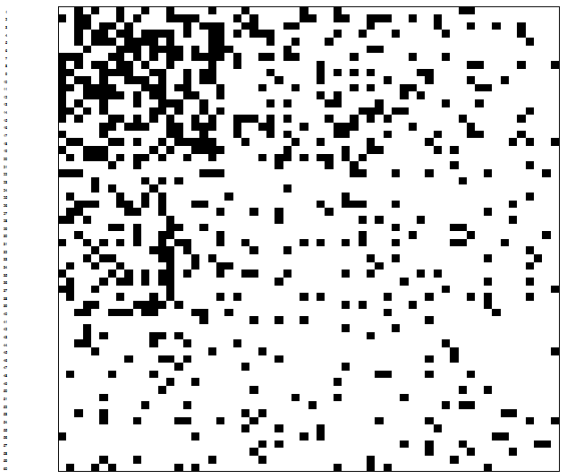
Slika 4.24: Primer generiranja omrežja *24*

Prikazana sta primera generiranja omrežja, pri katerih izhajamo iz matrike BM_7. Na Sliki 4.23 je prikazano omrežje z enakim številom enot v posameznem razbitju z različnima vrednostima polnih blokov (0.5 in 0.3). Na Sliki 4.24 je prikazano omrežje z različnim številom enot v posameznem razbitju. Polni blok z največjim številom enot ima višjo vrednost verjetnosti (0.5), medtem ko ima manjši polni blok manjšo vrednost (0.3).

$$BM_8 = \begin{bmatrix} \text{com} & \text{com} & \text{null} \\ \text{com} & \text{null} & \text{null} \\ \text{null} & \text{null} & \text{null} \end{bmatrix}$$

Bločni model BM_8 je sestavljen iz treh polnih ter šestih praznih blokov. Iz takega bločnega modela izhajamo pri generiranju omrežij *25* in *26*.

Slika 4.25: Primer generiranja omrežja *25*



Slika 4.26: Primer generiranja omrežja *26*

Predstavljena sta primera načina generiranja omrežij, pri katerih izhajamo iz bločne matrike BM_8. Na Sliki 4.25 je prikazan primer omrežja, kjer je število enot v posameznem razbitju enako. En polni blok ima vrednost verjetnosti 0.5, preostala dva pa vrednost 0.3. V drugem primeru (Slika 4.26) gre za različno število enot v posameznem razbitju. Blok z največjim številom enot je poln in ima vrednost verjetnosti 0.5, preostala dva polna bloka imata manjše število enot v razbitju in vrednost verjetnosti 0.3.

Bolj natančne lastnosti in ostali parametri so predstavljeni v nadaljevanju v Tabeli 4.2, kjer so predstavljene lastnosti po posameznem načinu generiranja omrežij.

Tabela 4.2: Lastnosti posameznega načina generiranja omrežij

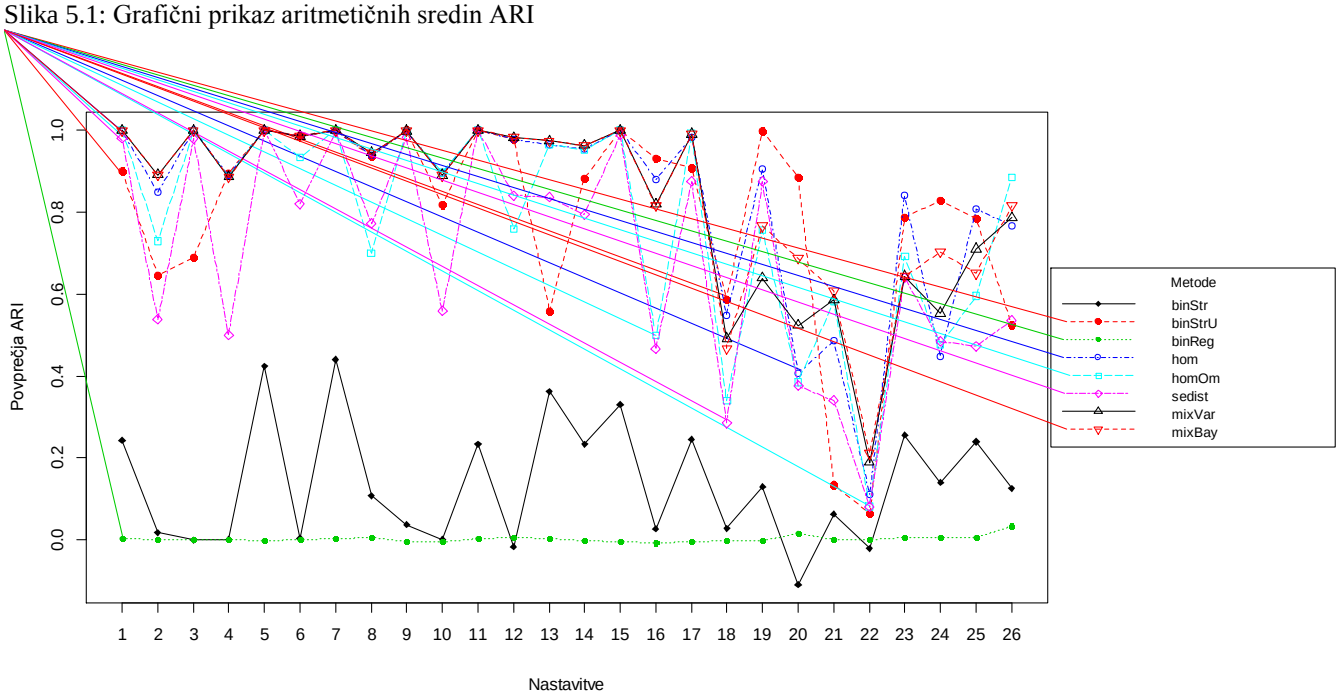
Generiranje omrežja	Število skupin	Bločni model	Velikosti skupin	Vrednosti verjetnosti po povezavah	
1	2	$\begin{bmatrix} null & com \\ com & com \end{bmatrix}$	c(30, 30)	0.5	0.1
2	2		c(30, 30)	0.3	0.1
3	2	$\begin{bmatrix} null & com \\ null & null \end{bmatrix}$	c(30, 30)	0.5	0.1
4	2		c(30, 30)	0.3	0.1
5	2	$\begin{bmatrix} com & null \\ null & com \end{bmatrix}$	c(30, 30)	0.5	0.1
6	2		c(30, 30)	0.3	0.1
7	2	$\begin{bmatrix} null & com \\ com & com \end{bmatrix}$	c(40, 20)	0.5	0.1
8	2		c(40, 20)	0.3	0.1
9	2	$\begin{bmatrix} null & com \\ null & null \end{bmatrix}$	c(40, 20)	0.5	0.1
10	2		c(40, 20)	0.3	0.1
11	2	$\begin{bmatrix} com & null \\ null & com \end{bmatrix}$	c(40, 20)	0.5	0.1
12	2		c(40, 20)	0.3	0.1
13	2	$\begin{bmatrix} com & com \\ com & null \end{bmatrix}$	c(30, 30)	$\begin{bmatrix} 0.5 & 0.3 \\ 0.3 & 0.1 \end{bmatrix}$	
14	2	$\begin{bmatrix} com & com \\ com & null \end{bmatrix}$	c(40, 20)	$\begin{bmatrix} 0.5 & 0.3 \\ 0.3 & 0.1 \end{bmatrix}$	
15	3	$\begin{bmatrix} com & null & null \\ null & com & null \\ null & null & com \end{bmatrix}$	c(20, 20,20)	0.5	0.1
16	3		c(20, 20, 20)	0.3	0.1
17	3	$\begin{bmatrix} com & com & com \\ com & com & null \\ com & null & null \end{bmatrix}$	c(20, 20, 20)	0.5	0.1
18	3		c(20, 20, 20)	0.3	0.1
19	3	$\begin{bmatrix} com & null & null \\ null & com & null \\ null & null & com \end{bmatrix}$	c(40, 10, 10)	0.5	0.1
20	3		c(40, 10, 10)	0.3	0.1
21	3	$\begin{bmatrix} com & com & com \\ com & com & null \\ com & null & null \end{bmatrix}$	c(40,10,10)	0.5	0.1
22	3		c(40,10,10)	0.3	0.1
23	3	$\begin{bmatrix} com & null & null \\ null & com & null \\ null & null & null \end{bmatrix}$	c(20,20,20)	$\begin{bmatrix} 0.5 & 0.1 & 0.1 \\ 0.1 & 0.3 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}$	
24	3	$\begin{bmatrix} com & null & null \\ null & com & null \\ null & null & null \end{bmatrix}$	c(40,10,10)	$\begin{bmatrix} 0.5 & 0.1 & 0.1 \\ 0.1 & 0.3 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}$	
25	3	$\begin{bmatrix} com & com & null \\ com & null & null \\ null & null & null \end{bmatrix}$	c(20,20,20)	$\begin{bmatrix} 0.5 & 0.3 & 0.1 \\ 0.3 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}$	
26	3	$\begin{bmatrix} com & com & null \\ com & null & null \\ null & null & null \end{bmatrix}$	c(40,10,10)	$\begin{bmatrix} 0.5 & 0.3 & 0.1 \\ 0.3 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}$	

5 REZULTATI

5.1 Primerjava in ocenjevanje metod

S popravljenim Randovim indeksom (ARI) primerjamo razbitja, ki jih poda posamezna metoda s pravim razbitjem, torej tistim, ki je uporabljeno pri generiranju omrežja. Med sabo primerjamo osem različnih metod neposrednega (*binStr*, *binStrU*, *binReg*, *hom*, *homOm*) in posrednega pristopa (*sedist*) za bločno modeliranje ter metodi za stohastično bločno modeliranje (*mixVar* in *mixBay*). Popravljen Randov indeks smo vzeli kot mero ustreznosti oz. kvalitete razbijta, ki ga poda določena metoda, s tem pa tudi kvaliteto metode.

Slika 5.1: Grafični prikaz aritmetičnih sredin ARI



S pomočjo rezultatov aritmetične sredine popravljenega Randovega indeksa, ki so prikazani v Sliki 5.1, in tabele z aritmetičnimi sredinami popravljenega Randovega indeksa (glej Priloga B – Tabela B.1) lahko ocenimo posamezne metode bločnega modeliranja in jih primerjamo med seboj.

Metode binarnega bločnega modeliranja: Pri metodah binarnega bločnega modeliranja primerjamo tri metode, ki smo jih poimenovali *binStr*, *binStrU* in *binReg*. Glede na aritmetične sredine ARI (glej Priloga B – Tabela B.1) ter grafične predstavitve podatkov (Slika 5.1) se izmed teh treh metod najbolj obnese pristop, kjer dovoljujemo bloke strukturne enakovrednosti z različno uteženimi bloki (*binStrU*). Preostali dve metodi se na teh omrežjih

slabo obneseta. Metoda *binStr*, kjer dovoljujemo bloke strukturne enakovrednosti, se najslabše obnese pri načinu generiranju omrežij *20*. Pri pristopu, kjer dovoljujemo bloke regularne enakovrednosti (*binReg*), so aritmetične sredine ARI na vseh generiranih omrežjih najnižje in približno enake, povsod so okoli 0. Vrednost 0 pomeni tako ujemanje, kot če bi bile skupine takih velikosti generirane slučajno. Nižji kot je ARI, manjše je ujemanje dveh razvrstitev (Rand 1971; Hubert in Arabie 1985). Prav ta metoda se izmed vseh ocenjevanih metod obnese najslabše.

Metodi homogenega bločnega modeliranja: Pri metodah homogenega bločnega modeliranja primerjamo dve metodi in opazimo, da se obe metodi glede na aritmetično sredino popravljenega Randovega Indeksa obneseta približno enako (*hom* in *homOm*). Nekoliko boljše se obnese metoda pristopa brez omejitev (*hom*). V primerjavi s preostalimi metodami imata metodi homogenega bločnega modeliranja visoke aritmetične sredine ARI (glej Priloga B – Tabela B.1).

Metoda posrednega pristopa: Metoda posrednega pristopa (*sedist*) se na večini omrežij obnese dobro, vendar vrednost popravljenega Randovega indeksa med različnimi generiranimi omrežji variira. Dokaj nizek popravljen Randov indeks je tako pri generiranju omrežij *18*, *20*, *21* in *22*, kjer je vrednost povprečja ARI 0.4 ali manj.

Metodi za stohastične modele: Iz grafa in tabele je razvidno, da se obe metodi stohastičnih modelov (*mixVar* in *mixBay*) na generiranih omrežjih obneseta dobro in tudi približno enako, saj so aritmetične sredine med metodama na istih omrežjih zelo podobne (glej Priloga B – Tabela B.1). Nekoliko bolje se obnese metoda *mixBay*, saj so povprečja ARI na večini omrežij 0.7 ali več. V primerjavi s preostalimi metodami se metodi za stohastične modele obneseta najbolje.

Če med sabo primerjamo vse metode, je iz rezultatov razvidno, da se na večini omrežjih najbolje obnese metoda za stohastične modele *mixBay*, sledita pa ji metoda za homogeno bločno modeliranje (*hom*) ter druga metoda za stohastične modele (*mixVar*). Dobro se obnese tudi metoda binarnega bločnega modeliranja, kjer dovoljujemo bloke strukturne enakovrednosti z različno uteženimi bloki (*binStrU*). Izmed vseh metod se najslabše obnese metoda binarnega bločnega modeliranja (*binReg*), kjer dovoljujemo bloke regularne enakovrednosti. Iz Tabele B.1 je razvidno, da so pri tej metodi vrednosti aritmetične sredine

ARI na vseh omrežjih najnižje in povsod približno enake vrednosti. Slabo se obnese tudi metoda binarnega bločnega modeliranja, kjer dovoljujemo bloke strukturne enakovrednosti (*binStr*), vrednosti povprečij ARI so 0.1 ali manj (glej Priloga B – Tabela B.1).

Opazimo, da se vse metode slabše obnesejo na generiranih omrežjih, kjer je vrednost verjetnosti polnega bloka manjša (0.3) v primerjavi z rezultati na povezavah polnih blokov z vrednostjo 0.5. Vse metode se slabo obnesejo pri načinu generiranja omrežij *22*. V tem primeru gre za bločni model, ki ima šest polnih blokov z enakim številom enot po skupinah ter vrednostjo verjetnosti v polnih blokih 0.3.

5.2 Primerjava vpliva različnih spremenljivk

Primerjali smo vpliv števila skupin, vrste bločnega modela, velikosti skupin ter verjetnosti po povezavah za polne bloke na aritmetične sredine popravljenega Randovega indeksa za osem različnih metod (*binStr*, *binStrU*, *binReg*, *hom*, *homOm*, *sedist*, *mixVar*, *mixBay*). V bolj podrobni analizi nas zanima predvsem, kdaj se metode obnesejo boljše in kdaj slabše, predvsem glede na posamezni vhodni parameter generiranega omrežja oz. posamezno spremenljivko.

Vrednosti spremenljivke bločni model smo zaradi kompleksnosti in lažje analize združili v tri skupine. Modele, kjer se nakazuje vrsta bločnega modela center-periferija, smo združili v skupino *CP*. Modele, kjer se nakazuje vrsta bločnega modela kohezivni bločni model, smo združili v skupino *Hom*. Model z enim samim polnim blokom oz. eno periferijo je v skupini *1P*. Vrednosti spremenljivke bločni model in ostalih spremenljivk so predstavljene v Tabeli 5.1.

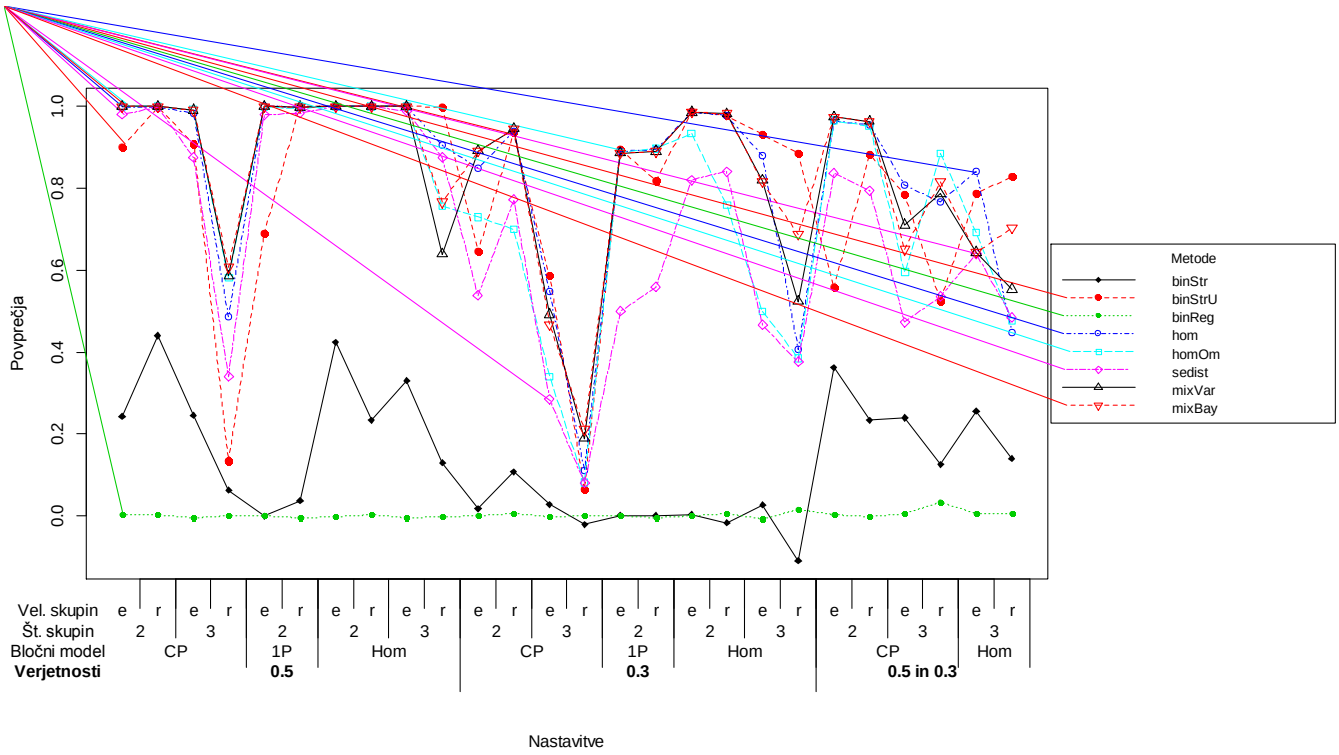
Tabela 5.1: Vrednosti neodvisnih spremenljivk

	Vrednosti
Verjetnosti po povezavah za polne bloke	*0.5* – 0.5, *0.3* – 0.3, *0.3 in 0.5* – 0.5 in 0.3
Bločni model	*CP* – BM_1, BM_5, BM_6, BM_8, *1P* – BM_2, *Hom* – BM_3, BM_4, BM_7
Število skupin	*2* – dve skupini, *3* – tri skupine

Velikosti skupin	*e* – skupine enake velikosti , *r* – skupine različnih velikosti

5.2.1 Metode glede na verjetnosti povezav

Slika 5.2: Grafični prikaz aritmetičnih sredin ARI glede na verjetnosti povezav



Primerjali smo metode glede na verjetnosti povezav, ki smo jih določili na polnih blokih. Opazujemo, kako se obnesejo posamezne metode v treh primerih: ko je vrednost polnega bloka 0.5, 0.3 in ko smo na polnih blokih uporabili obe verjetnosti. Glede na grafični prikaz vpliva verjetnosti povezave na aritmetične sredine popravljenega Randovega indeksa (Slika 5.2) ugotovimo, da se večina metod najbolje obnese tam, kjer smo uporabili verjetnost za polne bloke 0.5. Slabše se metode obnesejo tam, kjer je bila za polni blok uporabljena manjša vrednost (0.3).

Pri vrednosti 0.5 se najbolje obnesejo metode *hom*, *mixBay* in metoda *homOm*. Visoka povprečja ARI so tudi pri metodah *mixVar* in *sedist*. Večina povprečij je pri teh metodah 0.9 ali blizu te vrednosti, izstopajo pa vrednosti povprečja pri načinu generiranja omrežja *21*, kjer so te vrednosti opazno manjše. Gre za generiranje omrežij, kjer smo uporabili tri skupine različnih velikosti ter bločni model BM_5, kjer gre za bločni model center-periferija. Najvišja povprečja ARI, pri vrednosti 0.5, so pri načinu generiranja omrežij *5* in *7*, ki jima je skupno to, da smo tukaj uporabili dve skupini.

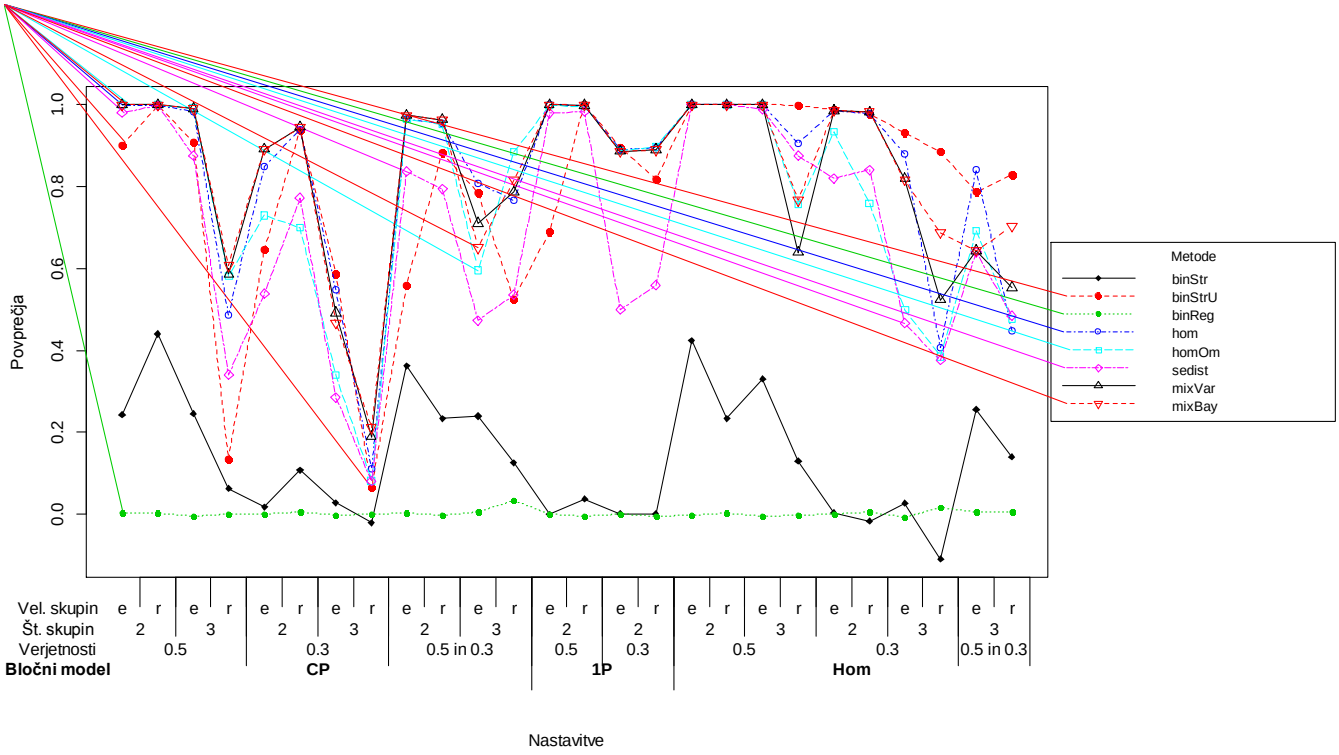
Pri vrednosti 0.3 se najbolje obneseta metodi za stohastične modele (*mixBay* in *mixVar*) ter metoda binarnega bločnega modeliranja, kjer dovoljujemo bloke strukturne enakovrednosti z različno uteženimi bloki *binStrU*. Najvišja povprečja ARI so pri generiranju omrežij *12* in *6*. Gre za generiranje omrežij s po dvema skupinama in bločnim modelom BM_3, pri katerem gre za kohezivni bločni model. Pri omrežjih z vrednostjo za polne bloke 0.3 se večina omrežij najslabše obnese na generiranju omrežij *22*, kjer smo uporabili enake parametre kot pri generiranju omrežij *21*.

Kjer smo uporabili obe vrednosti (0.3 in 0.5) za polne bloke na povezavah, se dobro obneseta metodi *hom* in *mixBay* ter, sicer nekoliko slabše, metodi *mixVar* in *homOm*. Višja povprečja ARI pri metodi *hom* in pri metodah za stohastične modele (*mixBay* in *mixVar*) so v tem primeru so pri generiranju omrežij *13* in *14*. Tukaj smo uporabili dve skupini in bločni model BM_6, kjer se nakazuje vrsta bločnega modela center-periferija.

Opazimo tudi, da se metoda *binStr* najslabše obnese v skupini, kjer je uporabljena manjša vrednost za polni blok (0.3). Metoda *binReg* se najslabše obnese v skupini, kjer smo za polni blok uporabili vrednost 0.5. Gre za metodi, ki se sicer izmed vseh metod obneseta najslabše.

5.2.2 Metode glede na bločni model

Slika 5.3: Grafični prikaz aritmetičnih sredin ARI glede na bločni model



Opazujemo, kako se posamezne metode obnesejo glede na vrsto bločnega modela, ki smo ga uporabili. Vrste bločnih modelov smo zaradi lažje analize združili v tri skupine, ki smo jih poimenovali *CP*, *1P* in *Hom* (glej Tabela 5.1). V skupini *CP* imamo bločne modele, kjer se nakazuje bločni model kohezivnih skupin, v skupini *Hom* so bločni modeli, kjer se nakazuje kohezivni bločni model, v skupini 1P pa je bločni model z enim samim polnim blokom. Opazimo, da se ocenjevane metode najboljše obnesejo v skupinah *1P* in *Hom*.

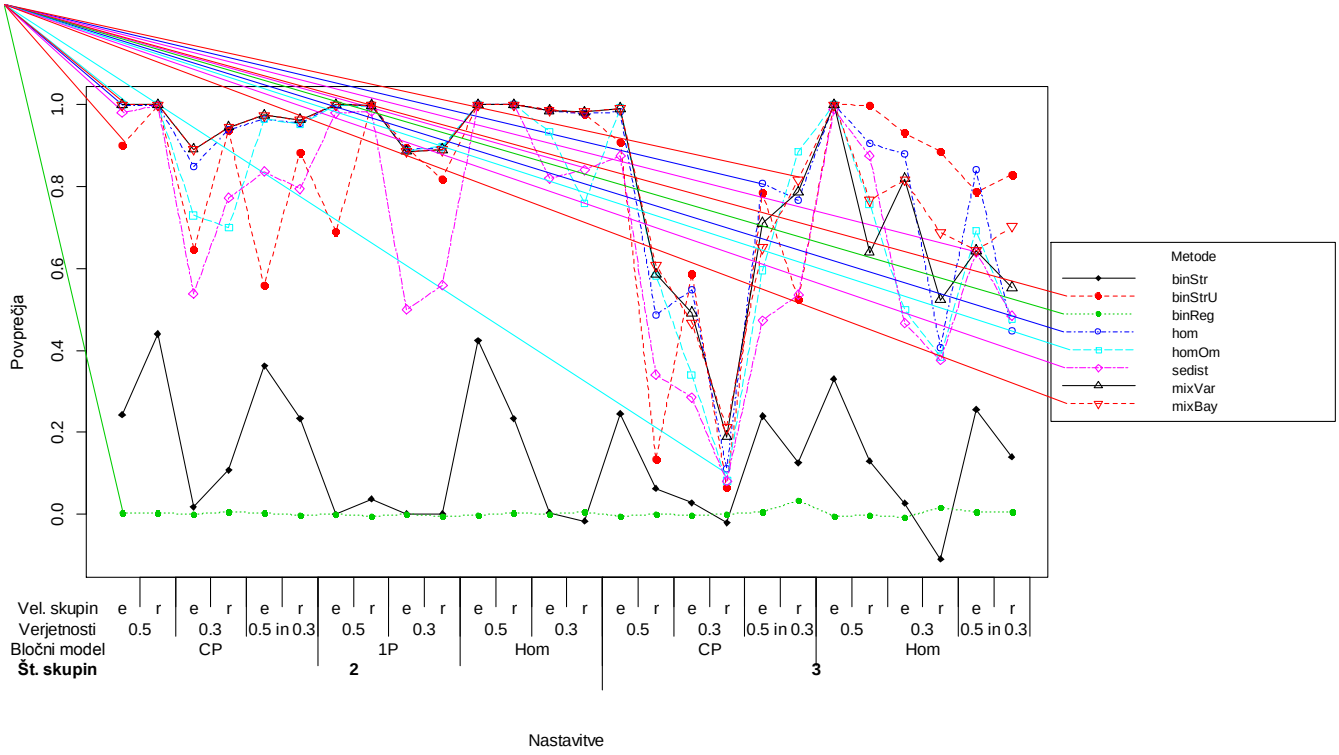
V skupini *CP*, kjer smo uporabili bločne modele center-periferija, se najboljše obneseta metodi za stohastično bločno modeliranje (*mixVar* in *mixBay*), sledita pa jima metodi *hom* in *homOm*. Nizka povprečja ARI v tej skupini opazimo predvsem pri generiranju omrežij *18* in *22*, kjer gre za omrežja s po tremi skupinami z različnim številom enot v posameznem razbitju in verjetnosmi za polne bloke 0.3. Nizka povprečja ARI opazimo tudi pri generiranju omrežij *21*, kjer je vrednost za polne bloke 0.5. Pri vseh treh načinih generiranja gre za bločni model BM_5, kjer izhajamo iz bločne matrike s šestimi polnimi (com) in tremi praznimi (null) bloki. Višja povprečja ARI najdemo pri generiranju omrežij *1* in *7* pri bločnem modelu BM_1, ki ima tri polne in en prazen blok. Opazimo tudi, da so v tej skupini nekoliko manjša povprečja ARI pri generiranju omrežij s po tremi skupinami.

Tako kot pri ostalih dveh skupinah se te štiri metode (*mixVar*, *mixBay*, *hom*, *homOm*) približno enako obnesejo tudi v skupini *1P*. Pri skupini z bločnim modelom *1P* imamo generirana omrežja samo s po dvema skupinama in z enim samim polnim blokom. V primerjavi s skupino *CP* in *Hom* imata v tej skupini medoti *binReg* in *binStr* najnižja povprečja ARI (glej Priloga B – Tabela B.1).

V skupini *Hom*, kjer imamo kohezivne bločne modele s povezavami znotraj vsake skupine in brez povezav med skupinami (Doreian, Batagelj in Ferligoj 2005, 236), se najbolje obnese metoda *binStrU*. V skupini teh bločnih modelov se dobro obnese tudi metoda *mixBay*, visoka povprečja ARI pa so tudi pri metodah *hom* in *mixVar*. Nižja povprečja ARI v skupini bločnih modelov *Hom* opazimo pri generiranjih omrežij *20* in *16*, kjer gre za bločni model BM_4 s popolnoma simetrično matriko s tremi polnimi bloki po diagonali. Prav tako nižja povprečja ARI opazimo pri generiranju omrežij *23* in *24*, kjer gre za bločni model s tremi razbitji in dvema polnima blokoma po diagonali. Najvišje aritmetične sredine ARI so pri generiranju omrežij *11*, *15* in *5*. Pri generiranih omrežjih *5* in *11* gre za bločni model BM_3 s simetrično bločno matriko z dvema polnima blokoma po diagonali. Pri generiranju omrežij *15* pa gre za bločni model BM_4 s simetrično bločno matriko s po tremi polnimi bloki po diagonali.

5.2.3 Metode glede na število skupin

Slika 5.4: Grafični prikaz aritmetičnih sredin ARI glede na število skupin



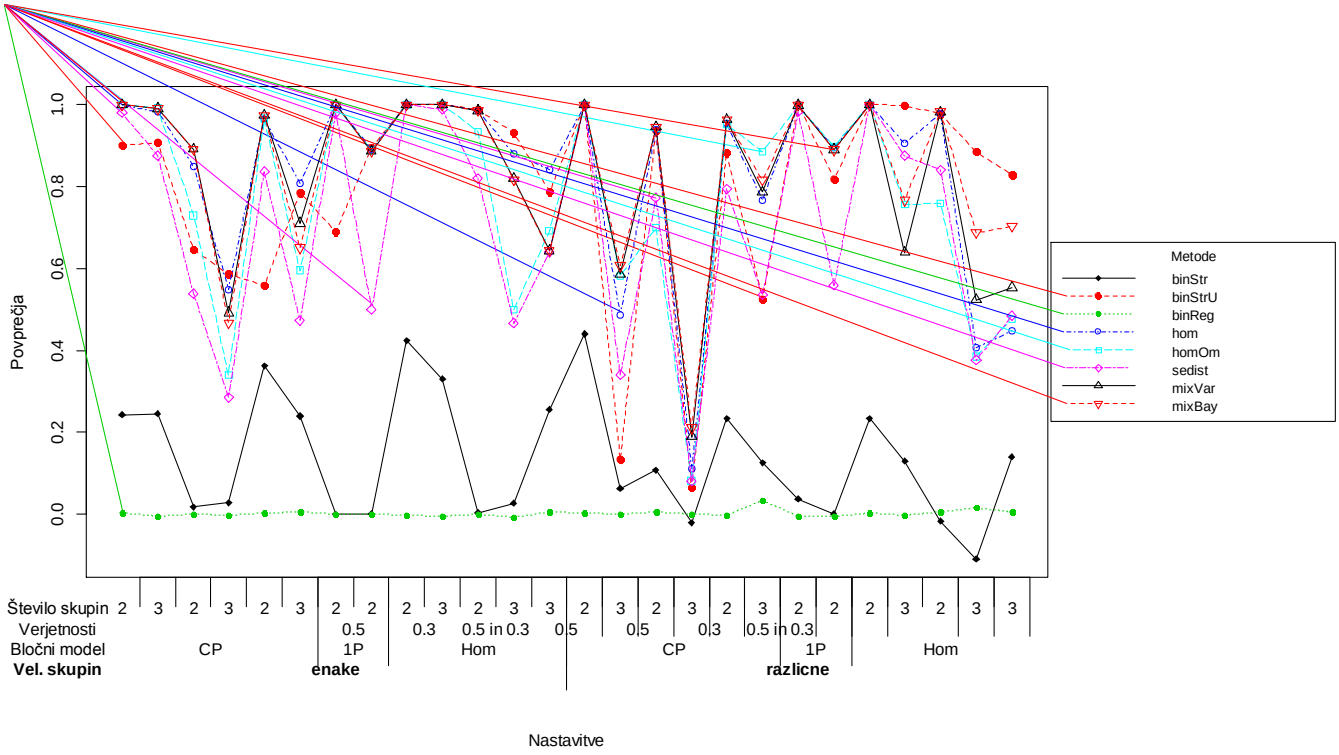
Med seboj primerjamo metode glede na število skupin. Primerjamo, kako se metode obnašajo, če gre za omrežje s po dvema ali tremi skupinami.

Iz grafičnega prikaza (Slika 5.4) je razvidno, da se metode boljše obnesejo na omrežjih s po dvema skupinama. Najboljše se obneseta metodi za stohastično bločno modeliranje (*mixVar* in *mixBay*) ter metodi za homogeno bločno modeliranje (*hom* in *homOm*). Pri teh štirih metodah so povprečja ARI blizu ali višja od 0.8. Najvišja povprečja so pri generiranju omrežij *7*, *5*, *11*, *1* in *9*. Tem načinom generiranja omrežja je skupno, da imajo vrednost verjetnosti polnega bloka 0.5. Nižja povprečja ARI opazimo na generiranih omrežjih z vrednostjo polnega bloka 0.3.

Pri omrežjih s po tremi skupinami se najbolje obnese metoda *binStrU*. Povprečja ARI so blizu 0.8 ali višja, manjša povprečja so pri generiranju omrežij *22*, *21*, *26* in *18*, kjer gre za omrežja, ki po bločnem modelu spadajo v skupino *CP*, kjer gre za model center-periferija. Metoda *binStrU* se najbolje obnese pri načinu generiranja omrežij *15*, *19*, *16* in *17*. Gre za omrežja, ki po bločnem modelu spadajo v skupino *Hom*, kjer gre za kohezivni bločni model. Visoka povprečja ARI pri omrežjih s po tremi skupinami so tudi pri metodah *mixBay*, *hom* in *mixVar*.

5.2.4 Metode po velikosti skupin

Slika 5.5: Grafični prikaz aritmetičnih sredin ARI glede na velikosti skupin



Primerjamo, kako se posamezne metode obnašajo glede na velikosti skupin. Primerjamo torej, kako se metode obnašajo, če je število enot v posameznem razbitju enako, in kako, če je število enot v posameznem razbitju različno.

Iz grafične predstavitve (Slika 5.5) in Tabele B.1 (Priloga B) s povprečji ARI je razvidno, da se metode nekoliko bolje obnesejo na generiranih omrežjih, kjer so skupine enake. Pri omrežjih, generiranih iz enakih skupin, se najboljše obnese metoda *hom*, kjer so aritmetične sredine ARI nad vrednostjo 0.8, razen pri generiranju omrežij *18*. Dobro pa se obneseta tudi metodi za stohastično bločno modeliranje *mixVar* in *mixBay*. Pri enakih skupinah imamo najnižja povprečja pri generiranju omrežij *18* in *25*. Gre za generiranje omrežij s po tremi skupinami in ki po bločnem modelu spadata v skupino *CP*. Najvišja povprečja so pri generiranju omrežij *15* in *5*, ki po bločnem modelu spadata v skupino *Hom* in imata pri polnem bloku vrednost na povezavah 0.5

Pri omrežjih, generiranih iz različnih skupin, se najbolje obnese metoda *mixBay*, vrednost povprečij je 0.7 ali več, izjemi sta pri generiranju omrežij *21* in *22*, kjer so povprečja ARI nižja. Visoka povprečja ARI imata tudi metodi *mixVar* in *binStrU*. Pri omrežjih z različnimi

skupinami imamo nižja povprečja pri generiranju omrežij *22*, *21*, *20* in *24*. Vsem je skupno, da gre za bločne matrike s po tremi skupinami. Višja povprečja pri različnih skupinah imamo pri generiranju omrežij *11*, *7* in *9*, kjer gre za generiranje omrežij s po dvema skupinama in vrednostjo na polnem bloku 0.5.

5.3 Vpliv spremenljivk na ARI

Za opazovanje vpliva posameznih spremenljivk smo uporabili analizo variance. Opazovali smo vpliv bločnega modela (spremenljivka, označena kot BM), števila skupin (SS), velikosti skupin (VS), verjetnosti po povezavah za polne bloke (VP) ter učinek metode (M). Vrednosti posameznih neodvisnih spremenljivk so opisane v Tabeli 5.1. Pri učinku metode opazujemo osem ocenjevanih metod (*binStr*, *binStrU*, *binReg*, *hom*, *homOm*, *sedist*, *mixVar*, *mixBay)*. V analizo variance so bile vključene neodvisne spremenljivke in vrednosti popravljenega Randovega indeksa na surovih podatkih, torej na vrednosti popravljenega Randovega indeksa za 2600 ponovitev posameznega generiranja omrežja oz. 20800 analiz.

Tabela 5.2: Analiza variance za ARI

Učinek	**Vsota kvadratov**	**Stopnje svobode (df)**	**F razmerje**	**Statistična značilnost**
M	2112.223	7	26396.255	2.20E-16
SS	197.28	1	17257.722	2.20E-16
VP	142.853	2	6248.29	2.20E-16
SS*VS	43.413	1	3797.736	2.20E-16
VS	25.072	1	2193.278	2.20E-16
BM*SS	22.493	1	1967.688	2.20E-16
BM	40.91	2	1789.364	2.20E-16
SS*VP	32.494	2	1421.24	2.20E-16
BM*SS*VS	13.775	1	1205.045	2.20E-16
BM*VP	38.81	3	1131.669	2.20E-16
SS*VS*VP	22.575	2	987.391	2.20E-16
SS*M	46.794	7	584.782	2.20E-16
VS*VP	11.472	2	501.787	2.20E-16
BM*VS*VP	15.218	3	443.735	2.20E-16
VP*M	55.057	14	344.022	2.20E-16
BM*M	42.018	14	262.551	2.20E-16
SS*VS*M	18.515	7	231.376	2.20E-16
BM*SS*VS*M	11.326	7	141.541	2.20E-16
SS*VP*M	22.204	14	138.739	2.20E-16
BM*SS*M	10.964	7	137.016	2.20E-16
BM*VP*M	28.396	21	118.287	2.20E-16

VS*M	8.663	7	108.256	2.20E-16
VS*VP*M	10.938	14	68.347	2.20E-16
BM*VS*M	9.71	14	60.673	2.20E-16
BM*SS*VP*M	4.702	7	58.758	2.20E-16
BM*VS*VP*M	14.018	21	58.395	2.20E-16
SS*VS*VP*M	7.413	14	46.318	2.20E-16
BM*SS*VS*VP	0.371	1	32.425	1.26E-05
BM*SS*VS*VP*M	1.342	7	16.775	2.20E-16
BM*SS*VP	0.088	1	7.717	2.20E-16
BM*VS	0.014	2	0.619	0.539
Reziduali	235.395	20592		

Legenda: BM – bločni model, SS – števili skupin, VS – velikosti skupin, VP – verjetnosti povezav za polne bloke, M – učinek metode

V Tabeli 5.2 so predstavljeni rezultati analize variance za popravljen Randov indeks. Predstavljeni so glavni učinki in vse interakcije med učinki (dvo-, tri-, štiri-, pet- smerni). Razvrščeni so glede na vrednost razmerja F, od večjih vrednosti proti manjšim.

Iz Tabele 5.2 je razvidno da so vsi učinki in vse interakcije med učinki statistično značilni, izjema je le interakcija med bločnim modelom in velikostjo skupin.

Najpomembnejši učinek je učinek metode, saj ima največjo vrednost vsote kvadratov (2112.223) in največjo vrednost F razmerja (26396.255). Drugi največji učinek je učinek števila skupin z vrednostjo vsote kvadratov 197.28 in vrednostjo F razmerja 17257.722. Glede na prejšnjo analizo in grafični prikaz vpliva števila skupin na aritmetično sredino ARI (Slika 5.1) opazimo, da so razlike med povprečji ARI med omrežji s po dvema skupinama in tremi skupinami dokaj opazne. Tretji največji učinek je učinek vrednosti povezav na polnih blokih. Četrti največji učinek je interakcija med številom in velikostjo skupin (Tabela 5.2). Učinek velikosti skupine ima nižji učinek kot v kombinaciji s številom skupin. Visoke vrednosti vsote kvadratov in F-razmerja imajo tudi učinek bločnega modela ter interakcijo bločnega modela s številom skupin.

6 ZAKLJUČEK

Cilj magistrskega dela je bil oceniti in primerjati različne metode bločnega modeliranja. Ugotavljali smo, katere metode se bolje obnesejo na posameznih vrstah omrežij. Ocenjevali in primerjali smo metode neposrednega in posrednega pristopa ter metodi za stohastične modele. Pri metodah neposrednega pristopa smo uporabili metode binarnega bločnega modeliranja (*binStr*, *binStrU* in *binReg*) ter metodi homogenega bločnega modeliranja (*hom*, *homOm*). Metoda posrednega pristopa je bila metoda *sedist*, ki računa razdaljo glede na strukturno enakovrednost (Lorrain in White 1971). Metodi za stohastično bločno modeliranje sta bili variacijska (*mixVar*) in bayesianska metoda (*mixBay*). Metode posrednega in neposrednega pristopa so implementirane v programu za analizo omrežij Pajek (Batagelj in Mrvar 2012) in v statističnem programu R v paketu blockmodeling (Žiberna 2010), metode za stohastične pristope pa so implementirane v R-paketku mixer (Ambroise in drugi 2013). Za naše delo smo uporabili statistični program R, kjer gre za celovito računalniško platformo in jezik za statistično analizo podatkov (Blejec 2009, 1). Uporabili smo že omenjena R-paketa blockmodeling (Žiberna 2010) in mixer (Ambroise in drugi 2013). Naša analiza je potekala na simuliranih omrežjih, pri katerih vhodne parametre določimo vnaprej. Vrste omrežij smo ločili glede na različne faktorje. Ločili smo jih predvsem glede na bločni model, število skupin v generiranem omrežju, velikosti skupin ter vrednosti verjetnosti na povezavah. Omrežja smo opazovali glede na vrednosti verjetnosti po povezavah na polnih (com) blokih. Posamezna simulacija je bila sestavljena s ponavljanjem več postopkov, ki so generiranje omrežij, izvajanje osmih različnih metod, iskanje 100 naključno izbranih razbitij ter primerjanje razbitij posamezne metode z razbitjem, uporabljenem pri generiranem omrežij s popravljenim Randovim indeksom. Vsaka simulacija je bila ponovljena 100-krat. Pri generiranju omrežij smo izhajali iz osmih različnih bločnih modelov in verjetnosti na povezavah za polne bloke. Omrežja smo generirali na 26 različnih načinov, njihove lastnosti pa so predstavljene v Tabeli 4.2.

Posamezne metode oz. njihova razbitja smo primerjali in ocenjevali s pomočjo popravljenega Randovega indeksa, ki je mera podobnosti med razvrstitvama (Rand 1971; Hubert in Arabie 1985, 198). Med seboj smo primerjali aritmetične sredine popravljenega Randovega indeksa posameznih metod, ki so prikazane v Prilogi B v Tabeli B.1 in na Sliki 5.1. Rezultati so pokazali, da se pri metodah binarnega bločnega modeliranja najbolje obnese metoda, kjer dovoljujemo bloke strukturne enakovrednosti z različno uteženimi bloki (*binStrU*), najslabše

pa metoda, kjer dovoljujemo bloke regularne enakovrednosti (*binReg*). Pri tej metodi so aritmetične sredine ARI na vseh generiranih omrežij najnižje in približno enake (glej Sliko 5.1). Metodi homogenega bločnega modeliranja se obneseta dobro na vseh vrstah omrežij, nekoliko bolje se obnese metoda homogenega bločnega modeliranja brez omejitev (*hom*). Pri metodah posrednega pristopa smo uporabili metodo *sedist*, ki se na večini omrežij obnese dobro, vendar vrednost aritmetične sredine ARI med različnimi omrežji variira. Metodi za stohastične modele (*mixVar* in *mixBay*) sta si glede na aritmetične sredine ARI na istih omrežjih zelo podobne (glej Priloga B – Tabelo B.1). Če med seboj primerjamo vse aritmetične sredine, opazimo, da se na večini omrežij najbolje obnese metoda za stohastične bločne modele *mixBay*, sledita pa ji metoda za homogeno bločno modeliranje (*hom*) ter druga metoda za stohastične modele (*mixVar*). Dobro se obnese tudi metoda binarnega bločnega modeliranja, kjer dovoljujemo bloke strukturne enakovrednosti z različno uteženimi bloki (*binStrU*). Najslabše izmed vseh metod se obnese metoda binarnega bločnega modeliranja (*binReg*), kjer dovoljujemo bloke regularne enakovrednosti. Vrednost aritmetičnih sredin ARI je na vseh omrežjih enaka in nizka, saj je na vseh omrežjih okoli 0, kar kaže na nizko ujemanje razbitja metode s pravim razbitjem, ki je bilo uporabljeno pri generiranju omrežja.
Če primerjamo naše rezultate z raziskavami, ki so bile že narejene, lahko ugotovimo, da dobimo podobne rezultate. Rezultati v Žiberna (2007) pokažejo, da se na simuliranih omrežjih, ki so sicer generirana na regularni enakovrednosti, najboljše obnese homogeno bločno modeliranje. Dobre rezultate dajo tudi pristopi za binarno in obteženo bločno modeliranje. Dobre rezultate pa so dali tudi pristopi za strukturno enakovrednost, torej homogeno bločno modeliranje za strukturno enakovrednost (Žiberna 2007, 179–180).

V našem magistrskem delu smo skušali tudi ugotoviti, na katerih vrstah omrežjih se posamezne metode najbolje obnesejo. Primerjali smo vpliv verjetnosti na povezavah, vrste bločnega modela, števila skupin in velikosti skupin na povprečja ARI posameznih metod. Pri spremenljivki verjetnosti po povezavah za polne bloke smo primerjali, ali se metode bolje obnesejo na omrežjih, kjer je vrednost polnega bloka 0.5, 0.3, ali tam kjer smo uporabili obe vrednosti. Metode so se najbolje obnesle tam, kjer smo uporabili vrednost 0.5, in slabše tam, kjer smo uporabili nižjo vrednost. Pri višji vrednosti so se najbolje obnesle metode *hom*, *mixBay* in metoda *homOm*. Pri omrežjih z vrednostjo polnega bloka 0.3, sta se najbolje obnesli metodi za stohastične bločne modele in metoda *binStrU*. Pri omrežjih z obema vrednostnima sta se dobro obnesli metodi *hom* in *mixBay*.

Pri spremenljivki bločni model smo zaradi lažje analize osem modelov razdelili v tri skupine: *CP*, *1P* in *Hom*. Metode so se pri bločnih modelih najbolje obnesle pri modelu z enim polnim blokom *1P* in skupini z bločnimi modeli s kohezivnimi bločnimi modeli *Hom.* Pri skupini *CP*, kjer smo uporabili bločne modele center-periferija, se najboljše obneseta metodi za stohastično bločno modeliranje in metodi za homogeno bločno modeliranje. Enako podobno se ti metodi obneseta v skupini bločnega modela *1P*. V skupini bločnih modelov *Hom* se je najbolje obnesla metoda *binStrU*. Pri spremenljivki števila skupin smo primerjali, če se metode bolje obnesejo na omrežjih s po dvema ali tremi skupinami. Iz Slike 5.3 je razvidno, da se bolje obnesejo na omrežjih s po dvema skupinama. Na takih omrežjih se najboljše obneseta metodi za stohastično bločno modeliranje (*mixVar* in *mixBay*) ter metodi za homogeno bločno modeliranje (*hom* in *homOm*). Pri omrežjih s po tremi skupinami, se najbolje obnese metoda *binStrU*. Če primerjamo metode glede na velikosti skupin (enake ali različne), se metode nekoliko bolje obnesejo na omrežjih z enakimi razbitji. Na takih omrežjih se najboljše obnesejo metoda *hom* ter metodi za stohastično bločno modeliranje. Pri omrežjih z različnimi skupinami pa se najboljše obnesejo metodi za stohastično bločno modeliranje in metoda *binStrU*.

Opazovali smo tudi vpliv učinka in interakcije učinkov posameznih spremenljivk, za kar smo uporabili analizo variance. Opazovali smo vpliv bločnega modela (spremenljivka, označena kot BM), števila skupin (SS), velikosti skupin (VS), verjetnosti po povezavah za polne bloke (VP) ter učinek metode (M). V analizo variance so bile vključene neodvisne spremenljivke in vrednosti popravljenega ARI na surovih podatkih, torej vrednosti ARI za 20800 analiz. Rezultati analize variance za glavne učinke in vse interakcije za popravljen Randov indeks so predstavljene v Tabeli 5.2. Vsi glavni učinki in interakcije med njimi so statistično značilne, razen interakcije med bločnim modelom in velikostjo skupin. Glavna ugotovitev je, da je najpomembnejši učinek, učinek metode, saj ima največjo vrednost vsote kvadratov in največjo vrednost F-razmerja. Visoke vrednosti le-teh so tudi pri učinku števila skupin, verjetnosti povezav in interakciji med številom skupin in velikostjo skupin. Najmanjši učinek ima interakcija med bločnim modelom in velikostjo skupin.

7 LITERATURA

Ambroise, Christophe, Gilles Grasseau, Mark Hoebeke, Pierre Latouche, Vicent Miele, Franck Picard in LAPACK authors. 2013. *mixer: Random graph clustering* (version 1.7). Dostopno prek: http://cran.rproject.org/web/packages/mixer/index.html (27. avgust 2014).

Ambroise, Christophe in Catherine Matias. 2012. New consistent and asymptotically normal parameter estimates for random-graph mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 74 (1): 3–35.

Analytics, Revolution in Steve Weston. 2014. *doParallel: Foreach parallel adaptor for the parallel package* (version 1.0.8). Dostopno prek: http://cran.r-project.org/web/packages/doParallel/index.html (27. avgust 2014).

Anderson, Carolyn J., Stanley Wasserman in Katherine Faust. 1992. Building stochastic blockmodels. *Special Issue on Blockmodels* 14 (1–2): 137–161.

Batagelj, Vladimir in Andrej Mrvar. 2011. *Pajek, Program for Analysis and Visualisation of Large Networks, Reference Manual, List of commands with short Explanation, version 2.05.* Dostopno prek: http://vlado.fmf.uni-lj.si/pub/networks/pajek/doc/pajekman.pdf /(1. december 2014).

Batagelj, Vladimir, Patrick Doreian in Anuška Ferligoj. 1992. An optimizational approach to regular equivalence. *Special Issue on Blockmodels* 14 (1–2): 121–135.

Batagelj, Vladimir, Anuška Ferligoj, in Patrick Doreian. 1992. Direct and Indirect Methods for Structural Equivalence. *Social Networks* 14 (1–2): 63–90.

Batagelj, Vladimir, Andrej Mrvar, Anuška Ferligoj in Patrick Doreian. 2004. Generalized blockmodeling with Pajek. *Metodološki zvezki* 1 (2): 455–467.

Blejec, Andrej. 2009. *Introduction to R.* Ljubljana.

Borgatti, Stephen P. in Martin P. Everett. 1992. Regular blockmodels of multiway, multimode matrices. *Social Networks* 14: 91–120.

Csardi G. in T. Nepusz. 2006. The igraph software package for complex network research. *InterJournal: Complex Systems.* Dostopno prek: http://igraph.org (27. avgust 2014).

Daudin, J. J., F. Picard, in S. Robin. 2008. A Mixture Model for Random Graphs. *Statistics and Computing* 18 (2): 173–183.

Dempster, A.P, N.M. Laird in D.B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39 (1): 1–38.

Doreian, Patrick, Vladimir Batagelj in Anuška Ferligoj. 1994. Partitioning networks based on generalized concepts of equivalence. *The Journal of Mathematical Sociology* 19 (1): 1–27.

--- 2005. *Generalized blockmodeling*. Structural analysis in the social sciences. New York: Cambridge University Press.

Ferligoj, Anuška. 1989. *Razvrščanje v skupine: teorija in uporaba v družboslovju / Anuška Ferligoj*. Zbirka Metodološki zvezki. Ljubljana: Fakulteta za sociologijo, politične vede in novinarstvo, Raziskovalni inštitut. Dostopno prek: http://dk.fdv.uni-lj.si/metodoloskizvezki/Pdfs/Mz_4Ferligoj.pdf (27. avgust 2014).

Ferligoj, Anuška in Vladimir Batagelj. 1996. Optimizacijski pristop k bločnim modelom. V *Slovenska država, družba in javnost*, uredil Anton Kramberger, 163–176. Ljubljana: Fakulteta za družbene vede.

Fienberg, Stephen E. in Stanley Wasserman. 1981. An Exponential Family of Probability Distributions for Directed Graphs: Comment. *Journal of the American Statistical Association* 76 (373): 54–57.

Florek, K., Lukaszewicz, J., Perkal, J., and Zubrzycki, S. 1951. Sur la Liaison et la Division des Points d'un Ensemble Fini. *Colloquium Mathematicae* 2: 282–285.

Gaujoux, Renaud. 2014. *doRNG: Generic Reproducible Parallel Backend for foreach Loops* (version 1.6). Dostopno prek: http://cran.r-project.org/web/packages/doRNG/index.html (27. avgust 2014).

Hlebec, Valentina in Tina Kogovšek. 2006. *Merjenje socialnih omrežij*. Študentska založba.

Holland, Paul W., Kathryn Blackmond Laskey in Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social Networks* 5 (2): 109–137.

Hubert, Lawrence in Phipps Arabie. 1985. Comparing Partitions. *Journal of Classification* 2 (1): 193–218.

Latouche, Pierre, Etienne Birmele in Christophe Ambroise. 2008. *Bayesian methods for graph clustering.* Dostopno prek: http://genome.jouy.inra.fr/ssb/preprint/SSB-RR-17.bayesianMixNet.pdf (27. avgust 2014).

--- 2012. Variational Bayesian inference and complexity control for stochastic block models. *Statistical Modelling* 12 (1): 93–115.

Lorrain, F. in H. C. White. 1971. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology* 1: 49–80.

McQuitty L. L. 1960. Hierarchical linkage analysis for the isolation of types. *Educational and Psychological Measurement* 20: 55–67.

--- 1966. Similarity Analysis by Reciprocal Pairs for Discrete and Continuous Data. *Educational and Psychological Measurement* 26: 825–831.

--- 1967. Expansion of similarity analysis by reciprocal pairs for discrete and continuous data. *Educational and Psychological Measurement* 27: 253–255.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, Friedrich Leisch, Chih-Chung Chang (libsvm C++-code), in Chih-Chen Lin (libsvm C++-code). 2014. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien* (version 1.6-3). Dostopno prek: http://cran.r-project.org/web/packages/e1071/index.html (27. avgust 2014).

Mizruchi, Mark S. 1994. Social Network Analysis: Recent Achievements and Current Controversies. *Acta Sociologica* 37 (4): 329–343.

de Nooy, Walter, Andrej Mrvar in Vladimir Batagelj. 2011. *Exploratory Social Network Analysis with Pajek. Revised and expanded second edition.* Cambridge, New York: Cambridge University Press.

Nowicki, Krzysztof in Tom A. B Snijders. 2001. Estimation and Prediction for Stochastic Blockstructures. *Journal of the American Statistical Association* 96 (455): 1077–1087.

Opsahl, Tore. 2012. *tnet: Software for Analysis of Weighted, Two-mode, and Longitudinal networks* (version 3.0.11). Dostopno prek: http://cran.r-project.org/web/packages/tnet/index.html (27. avgust 2014).

Picard, Franck, Vincent Miele, Jean-Jacques Daudin, Ludovic Cottret in Stéphane Robin. 2009. Deciphering the Connectivity Structure of Biological Networks Using MixNet. *BMC Bioinformatics* 10 (6): 1–11.

Rand, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association* 66: 846–850.

Revelle, William. 2014. *psych: Procedures for Psychological, Psychometric, and Personality Research* (version 1.4.8.11). Dostopno prek: http://cran.r-project.org/web/packages/psych/index.html (27. avgust 2014).

Sneath, P. H. A. 1957. The application of computers to taxonomy. *Journal of General Microbiology* 17 (1): 201–226.

Snijders, Tom A. B. in Krzysztof Nowicki. 1997. Estimation and Prediction for Stochastic Blockmodels for Graphs with Latent Block Structure. *Journal of Classification* 14 (1): 75–100.

Wasserman, Stanley in Carolyn Anderson. 1987. Stochastic a posteriori blockmodels: Construction and assessment. *Social Networks* 9 (1): 1–36.

Wasserman, Stanley in Katherine Faust. 1994. *Social Network Analysis: Methods and Applications*. Cambridge: Cambridge University Press.

White, Douglas R. in Karl P. Reitz. 1983. Graph and semigroup homomorphisms on networks of relations. *Social Networks* 5: 193–234.

White, Harrison C., Scott A. Boorman in Ronald L. Breiger. 1976. Social Structure from Multiple Networks. I. Blockmodels of Roles and Positions. *American Journal of Sociology* 81 (4): 730–780.

Yeung, KaYee in Walter L. Ruzzo. 2001. Details of the Adjusted Rand index and Clustering algorithms. Supplement to the paper "An empirical study on Principal Component Analysis for clustering gene expression data". Bioinformatics. 17: 763–774.

Zanghi, Hugo, Christophe Ambroise in Vincent Miele. 2008. Fast online graph clustering via Erdös-Rényi mixture. *Pattern Recogn* 41 (12): 3592–3599.

Žiberna, Aleš. 2007. Generalized blockmodeling of valued networks = (Posplošeno bločno modeliranje omrežij z vrednostmi na povezavah). Doktorska disertacija. Fakulteta za družbene vede: Ljubljana.

--- 2010. *blockmodeling: An R package for Generalized and classical blockmodeling of valued networks* (version 0.1.8). Dostopno prek: http://cran.at.r-project.org/web/packages/blockmodeling/index.html (27. avgust 2014).

--- 2009. Evaluation of direct and indirect blockmodeling of regular equivalence in valued networks by simulations .*Metodološki zvezki* 6 (2): 99–134.

--- 2013. Generalized blockmodeling of sparse networks. *Metodološki zvezki* 10 (2): 99–119.

Priloga A: Programska koda, uporabljena v magistrskem delu

```
################################################################################
#######
#R version 3.0.0 (2013-04-03) -- "Masked Marvel"
#Copyright (C) 2013 The R Foundation for Statistical Computing
#Platform: i386-w64-mingw32/i386 (32-bit)
#######
################################################################################
#######

###Generiranje podatkov

### število ponovitev m<-100
### velikost skupin cca. n<-60
### število iskanja razbitij rep = 100

################################################################################
#######PACKAGES
#install.packages(c("e1071","mixer","doParallel","doRNG","tnet","igraph", "psych"))
install.packages("blockmodeling",repos="http://R-Forge.R-project.org")

################################################################################
#######LIBRARY
library(blockmodeling)
library(e1071)
library(mixer)
library(doParallel)
library(doRNG)
library(tnet)
library(igraph)
#######
################################################################################

#### GENERIRANJE IN SIMULACIJA PODATKOV ##################################
####### dve skupini (30:30)

########### Generiranje podatkov 1
## Bločni model:
## null com
## com  com

o<-60
n<-c(30,30)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.5)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
plot.mat(net)

## Simulacija podatkov 1
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_1<-matrix(NA,ncol=8,nrow=m)
colnames(res_1)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(30,30)
k<-length(n) #število skupin
clu<-rep(1:k,times=n) #pripadnost skupini
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.5)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
```

```
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_1[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_1[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_1[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_1[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_1[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_1[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_1[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_1[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_1,2,mean)
apply(res_1,2,sd)

########### Generiranje podatkov 2
## Bločni model:
## null com
## com  com

o<-60
n<-c(30,30)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
plot.mat(net)

## Simulacija podatkov 2
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_2<-matrix(NA,ncol=8,nrow=m)
colnames(res_2)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(30,30)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)

  d<-sum(net)/prod(dim(net))
```

```
  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_2[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_2[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_2[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_2[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_2[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_2[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_2[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_2[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_2,2,mean)
apply(res_2,2,sd)
save.image()

########### Generiranje podatkov 3
## Bločni model:
## null com
## null null

n<-c(30,30)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 3
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_3<-matrix(NA,ncol=8,nrow=m)
colnames(res_3)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(30,30)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)

  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_3[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]
```

```
  resBinStrStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_3[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_3[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_3[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_3[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_3[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_3[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_3[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_3,2,mean)
apply(res_3,2,sd)

########### Generiranje podatkov 4
## Bločni model:
## null com
## null null

o<-60
n<-c(30,30)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 4
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_4<-matrix(NA,ncol=8,nrow=m)
colnames(res_4)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(30,30)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)

  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_4[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_4[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]
```

```
  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_4[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_4[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_4[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_4[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_4[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_4[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_4,2,mean)
apply(res_4,2,sd)

############ Generiranje podatkov 5
## Bločni model:
## com null
## null com

n<-c(30,30)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
plot.mat(net)

## Simulacija podatkov 5
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_5<-matrix(NA,ncol=8,nrow=m)
colnames(res_5)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(30,30)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)

  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_5[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_5[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_5[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]
```

```
  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_5[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_5[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_5[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_5[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_5[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_5,2,mean)
apply(res_5,2,sd)

############ Generiranje podatkov 6
## Bločni model:
## com null
## null com

n<-c(30,30)
net<-matrix(NA,ncol=n,nrow=n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
plot.mat(net)

## Simuliranje podatkov 6
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_6<-matrix(NA,ncol=8,nrow=m)
colnames(res_6)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(30,30)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)

  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_6[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_6[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_6[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_6[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]
```

```
  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_6[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_6[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_6[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_6[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_6,2,mean)
apply(res_6,2,sd)

####### dve skupini (40:20)

############ Generiranje podatkov 7
## Bločni model:
## null com
## com  com

o<-60
n<-c(40,20)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.5)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
plot.mat(net)

## Simulacija podatkov 7
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_7<-matrix(NA,ncol=8,nrow=m)
colnames(res_7)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.5)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)

  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_7[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_7[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_7[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_7[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]
```

```
  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_7[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_7[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_7[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_7[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_7,2,mean)
apply(res_7,2,sd)

############ Generiranje podatkov 8
## Bločni model:
## null com
## com  com

n<-c(40,20)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
plot.mat(net)

## Simulacija 8
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_8<-matrix(NA,ncol=8,nrow=m)
colnames(res_8)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)

  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_8[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_8[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_8[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_8[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_8[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]
```

```
  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_8[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_8[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_8[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_8,2,mean)
apply(res_8,2,sd)

####### dve skupini (40:20)

############ Generiranje podatkov 9
## Bločni model:
## null com
## null null

n<-c(40,20)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
plot.mat(net)

## Simulacija 9
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_9<-matrix(NA,ncol=8,nrow=m)
colnames(res_9)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-(c(40,20)
k<-length(n)
clu<-rep(1:2,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)

  d<-sum(net)/prod(dim(net))

  resBinStrStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_9[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_9[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_9[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_9[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_9[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
```

```
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_9[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_9[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_9[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_9,2,mean)
apply(res_9,2,sd)

############ Generiranje podatkov 10
## Bločni model:
## null com
## null null

n<-c(40,20)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 10
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_10<-matrix(NA,ncol=8,nrow=m)
colnames(res_10)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.1)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)

  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_10[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_10[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_10[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_10[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_10[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_10[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]
```

```
  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_10[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_10[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_10,2,mean)
apply(res_10,2,sd)

############ Generiranje podatkov 11
## Bločni model:
## com  null
## null com

o<-60
n<-c(40,20)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
plot.mat(net)

## Simulacija podatkov 11
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_11<-matrix(NA,ncol=8,nrow=m)
colnames(res_11)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_11[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_11[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_11[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_11[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_11[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_11[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
```

```
  res_11[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_11[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_11,2,mean)
apply(res_11,2,sd)

############ Generiranje podatkov 12
## Bločni model:
## com  null
## null com

n<-c(40,20)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
plot.mat(net)

## Simulacija podatkov 12
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_12<-matrix(NA,ncol=8,nrow=m)
colnames(res_12)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_12[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_12[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_12[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_12[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_12[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_12[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_12[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
```

```
  res_12[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_12,2,mean)
apply(res_12,2,sd)

####### dve skupini (30:30) verjetnosti po povezavah

############ Generiranje podatkov 13
## Bločni model:
## com com
## com null

o<-60
n<-c(30,30)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 13
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_13<-matrix(NA,ncol=8,nrow=m)
colnames(res_13)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(30,30)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_13[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_13[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_13[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_13[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_13[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_13[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_13[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_13[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
```

```
}

apply(res_13,2,mean)
apply(res_13,2,sd)

####### dve skupini (40:20) verjetnosti po povezavah

############ Generiranje podatkov 14
## Bločni model:
#com com
#com null

o<-60
n<-c(40,20)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 14
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_14<-matrix(NA,ncol=8,nrow=m)
colnames(res_14)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,20)
k<-length(n)
clu<-rep(1:2,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_14[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_14[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_14[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_14[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_14[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_14[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_14[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_14[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}
```

```
apply(res_14,2,mean)
apply(res_14,2,sd)

####### tri skupine (20:20:20)

############ Generiranje podatkov 15
## Bločni model:
## com  null null
## null com  null
## null null com

o<-60
n<-c(20,20,20)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.5)
plot.mat(net)

## Simulacija podatkov 15
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_15<-matrix(NA,ncol=8,nrow=m)
colnames(res_15)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(20,20,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.5)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_15[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_15[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_15[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_15[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_15[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_15[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]
```

```
  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_15[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_15[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_15,2,mean)
apply(res_15,2,sd)

############ Generiranje podatkov 16
## Bločni model:
## com  null null
## null com  null
## null null com

n<-c(20,20,20)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
k<-length(n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.3)
plot.mat(net)

## Simulacija podatkov 16
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_16<-matrix(NA,ncol=8,nrow=m)
colnames(res_16)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(20,20,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.3)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_16[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_16[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_16[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_16[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
```

```
  res_16[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_16[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_16[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_16[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_16,2,mean)
apply(res_16,2,sd)

############ Generiranje podatkov 17
## Bločni model:
## com  com  com
## com  com  null
## com  null   null

n<-c(20,20,20)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.5)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.5)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.5)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 17
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_17<-matrix(NA,ncol=8,nrow=m)
colnames(res_17)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(20,20,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.5)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.5)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.5)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_17[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_17[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
```

```
  res_17[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_17[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_17[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_17[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_17[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_17[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_17,2,mean)
apply(res_17,2,sd)

############ Generiranje podatkov 18
## Bločni model:
## com  com  com
## com  com  null
## com  null   null

n<-c(20,20,20)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.3)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.3)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 18
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_18<-matrix(NA,ncol=8,nrow=m)
colnames(res_18)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(20,20,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.3)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.3)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
```

```
  res_18[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_18[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_18[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_18[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_18[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_18[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_18[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_18[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_18,2,mean)
apply(res_18,2,sd)

####### tri skupine (40:10:10)

############ Generiranje podatkov 19
## Bločni model:
## com null null
## null com null
## null null com

o<-60
n<-c(40,10,10)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
k<-length(n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.5)
plot.mat(net)

## Simulacija podatkov 19
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_19<-matrix(NA,ncol=8,nrow=m)
colnames(res_19)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,10,10)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
```

```
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.5)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_19[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_19[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_19[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_19[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_19[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_19[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_19[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_19[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_19,2,mean)
apply(res_19,2,sd)

############ Generiranje podatkov 20
## Bločni model:
## com  null null
## null  com  null
## null  null com

n<-c(40,10,10)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.3)
plot.mat(net)

## Simulacija podatkov 20
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_20<-matrix(NA,ncol=8,nrow=m)
colnames(res_20)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,10,10)
```

```
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.3)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_20[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_20[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_20[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_20[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_20[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_20[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_20[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_20[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_20,2,mean)
apply(res_20,2,sd)

############ Generiranje podatkov 21
## Bločni model:
## com com  com
## com com  null
## com null null

n<-c(40,10,10)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.5)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.5)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.5)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 21
set.seed(2010)
```

```
m<-100 # število ponovitev

# podatke shranjujemo sem
res_21<-matrix(NA,ncol=8,nrow=m)
colnames(res_21)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,10,10)
k<-length(n)
clu<-rep(1:k,times=n))
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.5)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.5)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.5)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.5)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.5)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_21[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_21[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_21[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_21[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_21[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_21[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_21[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_21[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_21,2,mean)
apply(res_21,2,sd)

############ Generiranje podatkov 22
## Bločni model:
## com com  com
## com com  null
## com null null

n<-c(40,10,10)
net<-matrix(NA,ncol=o,nrow=o)
clu<-rep(1:k,times=n)
k<-length(n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.3)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
```

```
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.3)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
plot.mat(net)

## Simuliranje podatkov 22
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_22<-matrix(NA,ncol=8,nrow=m)
colnames(res_22)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,10,10)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.3)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.3)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.3)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_22[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_22[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_22[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_22[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_22[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_22[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_22[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_22[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_22,2,mean)
apply(res_22,2,sd)

############ Generiranje treh skupin (40:10:10) verjetnosti po povezavah

############ Generiranje 23
## Bločni model:
## com null null
## null com null
## null null null

o<-60
n<-c(20,20,20)
```

```
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
plot.mat(net)

## Simuliranje podatkov 23
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_23<-matrix(NA,ncol=8,nrow=m)
colnames(res_23)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(20,20,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_23[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_23[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

  resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_23[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_23[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_23[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_23[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
  res_23[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_23[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_23,2,mean)
apply(res_23,2,sd)
```

```
### Generiranje treh skupin (40:10:10) verjetnosti po povezavah

############ Generiranje 24
## Bločni model:
## com null null
## null com null
## null null null

o<-60
n<-c(40,10,10)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
plot.mat(net)

## Simulacija podatkov 24
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_24<-matrix(NA,ncol=8,nrow=m)
colnames(res_24)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,10,10)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
  net<-matrix(NA,ncol=o,nrow=o)
  tclu<-table(clu)
  net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
  net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.1)
  net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.1)
  net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.3)
  net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
  net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
  net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
  net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)

  d<-sum(net)/prod(dim(net))

  resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
  res_24[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

  resBinStrStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
  res_24[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrStrU)))[[4]]

  resBinStrReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
  res_24[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

  resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
  res_24[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

  resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
  res_24[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

  D<-sedist(M=net)
  hcluWard<-hclust(D, method = "ward")
  cluWard<-cutree(hcluWard,k=k)
  res_24[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

  resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
```

```
  res_24[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

  resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
  resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
  res_24[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_24,2,mean)
apply(res_24,2,sd)

############ Generiranje 25
## Bločni model:
## com  com  null
## com  null null
## null null null

o<-60
n<-c(20,20,20)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
plot.mat(net)

## Simuliranje podatkov 25
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_25<-matrix(NA,ncol=8,nrow=m)
colnames(res_25)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(20,20,20)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
   net<-matrix(NA,ncol=o,nrow=o)
   tclu<-table(clu)
   net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
   net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
   net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
   net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
   net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
   net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
   net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
   net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
   net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
   d<-sum(net)/prod(dim(net))

   resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
   res_25[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

   resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
   res_25[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]

   resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
   res_25[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

   resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
   res_25[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

   resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
   res_25[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]
```

```
   D<-sedist(M=net)
   hcluWard<-hclust(D, method = "ward")
   cluWard<-cutree(hcluWard,k=k)
   res_25[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

   resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50,
improve=FALSE, verbose=TRUE)
   resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
   res_25[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

   resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
   resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
   res_25[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_25,2,mean)
apply(res_25,2,sd)

### Generiranje treh skupin (40:10:10) verjetnosti po povezavah

############ Generiranje 26
## Bločni model:
## com  com  null
## com  null  null
## null  null  null

o<-60
n<-c(40,10,10)
net<-matrix(NA,ncol=o,nrow=o)
k<-length(n)
clu<-rep(1:k,times=n)
tclu<-table(clu)
net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
plot.mat(net)

## Simuliranje podatkov 26
set.seed(2010)
m<-100 # število ponovitev

# podatke shranjujemo sem
res_26<-matrix(NA,ncol=8,nrow=m)
colnames(res_26)<-c("binStr","binStrU","binReg","hom","homOm", "sedist", "mixVar", "mixBay")

n<-c(40,10,10)
k<-length(n)
clu<-rep(1:k,times=n)
for(i in seq_len(m)){
   net<-matrix(NA,ncol=o,nrow=o)
   tclu<-table(clu)
   net[clu==1,clu==1]<-rbinom(n=tclu[1]*tclu[1],size=1,p=0.5)
   net[clu==1,clu==2]<-rbinom(n=tclu[1]*tclu[2],size=1,p=0.3)
   net[clu==1,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
   net[clu==2,clu==1]<-rbinom(n=tclu[2]*tclu[1],size=1,p=0.3)
   net[clu==2,clu==2]<-rbinom(n=tclu[2]*tclu[2],size=1,p=0.1)
   net[clu==2,clu==3]<-rbinom(n=tclu[2]*tclu[3],size=1,p=0.1)
   net[clu==3,clu==1]<-rbinom(n=tclu[3]*tclu[1],size=1,p=0.1)
   net[clu==3,clu==2]<-rbinom(n=tclu[3]*tclu[2],size=1,p=0.1)
   net[clu==3,clu==3]<-rbinom(n=tclu[3]*tclu[3],size=1,p=0.1)
   d<-sum(net)/prod(dim(net))

   resBinStr<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="bin")
   res_26[i,"binStr"]<-classAgreement(tab=table(clu,clu(resBinStr)))[[4]]

   resBinStrU<-optRandomParC(M=net, k=k, approach="bin", blocks=c("nul","com"), blockTypeWeights=c(nul=1,com=d/(1-d)),
rep=100, nCores = 3,useOptParMultiC=TRUE,printRep=50)
   res_26[i,"binStrU"]<-classAgreement(tab=table(clu,clu(resBinStrU)))[[4]]
```

```
   resBinReg<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","reg", "com"),approach="bin")
   res_26[i,"binReg"]<-classAgreement(tab=table(clu,clu(resBinReg)))[[4]]

   resSs<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks="com",approach="hom",homFun="ss")
   res_26[i,"hom"]<-classAgreement(tab=table(clu,clu(resSs)))[[4]]

   resSsLim<-optRandomParC(M=net, k=k,rep=100, nCores = 3,blocks=c("nul","com"),approach="hom",homFun="ss",
usePreSpecM=c(0,3), preSpecM=c(0,d))
   res_26[i,"homOm"]<-classAgreement(tab=table(clu,clu(resSsLim)))[[4]]

   D<-sedist(M=net)
   hcluWard<-hclust(D, method = "ward")
   cluWard<-cutree(hcluWard,k=k)
   res_26[i,"sedist"]<-classAgreement(tab=table(clu,cluWard))[[4]]

   resmixVar<-mixer( x=net, qmin=k, qmax=k, method="variational", directed = TRUE, nbiter=100, fpnbiter=50,
improve=FALSE, verbose=TRUE)
   resmixVarclu<-apply(resmixVar$output[[1]]$Taus,2,which.max)
   res_26[i,"mixVar"]<-classAgreement(tab=table(clu,resmixVarclu))[[4]]

   resmixBay<-mixer( x=net, qmin=k, qmax=k, method="bayesian", directed = TRUE, nbiter=100, fpnbiter=50, improve=FALSE,
verbose=TRUE)
   resmixBAyclu<-apply(resmixBay$output[[1]]$Taus,2,which.max)
   res_26[i,"mixBay"]<-classAgreement(tab=table(clu,resmixBAyclu))[[4]]
}

apply(res_26,2,mean)
apply(res_26,2,sd)

##############################################################################
####### ARI - aritmetična sredina
a<-apply(res_1,2,mean)
b<-apply(res_2,2,mean)
c<-apply(res_3,2,mean)
d<-apply(res_4,2,mean)
e<-apply(res_5,2,mean)
f<-apply(res_6,2,mean)
g<-apply(res_7,2,mean)
h<-apply(res_8,2,mean)
i<-apply(res_9,2,mean)
j<-apply(res_10,2,mean)
k<-apply(res_11,2,mean)
l<-apply(res_12,2,mean)
m<-apply(res_13,2,mean)
n<-apply(res_14,2,mean)
o<-apply(res_15,2,mean)
p<-apply(res_16,2,mean)
r<-apply(res_17,2,mean)
s<-apply(res_18,2,mean)
t<-apply(res_19,2,mean)
u<-apply(res_20,2,mean)
v<-apply(res_21,2,mean)
z<-apply(res_22,2,mean)
x<-apply(res_23,2,mean)
y<-apply(res_24,2,mean)
x1<-apply(res_25,2,mean)
y1<-apply(res_26,2,mean)

B<-rbind (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,r,s,t,u,v,z,x,y,x1,y1)
B ####### aritmetične sredine ARI

##############################################################################
####### ARI - standardni odklon
a1<-apply(res_1,2,sd)
b1<-apply(res_2,2,sd)
c1<-apply(res_3,2,sd)
d1<-apply(res_4,2,sd)
e1<-apply(res_5,2,sd)
f1<-apply(res_6,2,sd)
g1<-apply(res_7,2,sd)
h1<-apply(res_8,2,sd)
i1<-apply(res_9,2,sd)
j1<-apply(res_10,2,sd)
k1<-apply(res_11,2,sd)
l1<-apply(res_12,2,sd)
m1<-apply(res_13,2,sd)
```

```
n1<-apply(res_14,2,sd)
o1<-apply(res_15,2,sd)
p1<-apply(res_16,2,sd)
r1<-apply(res_17,2,sd)
s1<-apply(res_18,2,sd)
t1<-apply(res_19,2,sd)
u1<-apply(res_20,2,sd)
v1<-apply(res_21,2,sd)
z1<-apply(res_22,2,sd)
x1<-apply(res_23,2,sd)
y11<-apply(res_24,2,sd)
x11<-apply(res_25,2,sd)
y11<-apply(res_26,2,sd)

C<-rbind (a1,b1,c1,d1,e1,f1,g1,h1,i1,j1,k1,l1,m1,n1,o1,p1,r1,s1,t1,u1,v1,z1,x1,y1,x11,y11)
C ####### standardni odkloni ARI

podatki <- data.frame (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,r,s,t,u,v,z,x,y,x1,y1)
podatki

#############################################################################
############ Izvoz podatkov v excell ########################################

library(foreign)
write.csv2(x=B,file="podatki_mean_ari.csv") #### tabela z aritmetičnimi sredinami ARI
write.csv2(x=C,file="podatki_sd_ari.csv") #### tabela s standardnimi odkloni ARI

############ GRAFIČNI PRIKAZ ### ARI aritmetične sredine ####################
win.graph()
mar.def<-c(5,4,4,1)+0.1
par(mar=mar.def+c(4,0,0,14),mfrow=c(1,1),xpd=TRUE)
matplot(B,type="o",xaxt="n",ylab="Povprečja ARI", lty = 1:6, col = 1:6, pch = (18:25))
legend(x = "right",legend = colnames(B), ncol=1,inset = -0.3,0,title = "Metode" ,lty = 1:6, lwd = 1, pch = (18:25), col = 1:6,
cex=0.8)
title(xlab="Nastavitve",line=3.5)
axis(side=1,at=1:dim(B)[1])

############ GRAFIČNI PRIKAZ ### ARI standardni odkloni ######################
win.graph()
mar.def<-c(5,4,4,1)+0.1
par(mar=mar.def+c(0,0,0,14),mfrow=c(1,1),xpd=TRUE)
matplot(C,type="o",xaxt="n",ylab="Standardni odklon ARI",lty = 1:6, col = 1:6, pch = (18:25))
legend(x = "right",legend = colnames(C), ncol=1,inset = -0.3,0,title = "Metode" ,lty = 1:6, lwd = 1, pch = (18:25), col = 1:6,
cex=0.8)
title(xlab="Nastavitve",line=3.5)
axis(side=1,at=1:dim(C)[1])

###### GRAFIČNI PRIKAZ po posameznih spremenljivkah ##########################

stevilo_skupin<-factor(c(1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2),labels=c(2,3))## 1- dve skupini; 2- 3 skupine
blocni_model<-factor(c(1,1,2,2,3,3,1,1,2,2,3,3,6,6,4,4,5,5,4,4,5,5,7,7,8,8))## 1- 1.model itd.
blocni_model[blocni_model==4]<-3
blocni_model[blocni_model==6]<-1
blocni_model[blocni_model==7]<-3
blocni_model[blocni_model==8]<-1
blocni_model[blocni_model==5]<-1
blocni_model<-factor(blocni_model)
levels(blocni_model)<-c("CP","1P","Hom")

summary(blocni_model)
velikosti_skupin<-factor(c(1,1,1,1,1,1,2,2,2,2,2,2,1,2,1,1,1,1,2,2,2,2,1,2,1,2), labels=c("enake","razlicne"))## 1- enake velikosti,
2- različne velikosti
verjetnosti_povezav<-factor(c(1,2,1,2,1,2,1,2,1,2,1,2,3,3,1,2,1,2,1,2,1,2,3,3,3,3),labels=c(0.5, 0.3,"0.5 in 0.3"))
summary(velikosti_skupin)

tabela<-data.frame(metoda=B,
stevilo_skupin=stevilo_skupin,velikosti_skupin=velikosti_skupin,blocni_model=blocni_model,verjetnosti_povezav=verjetnosti_po
vezav)

tabela1<-tabela[order(verjetnosti_povezav,blocni_model,stevilo_skupin,velikosti_skupin),]
tabela2<-tabela[order(blocni_model,verjetnosti_povezav,stevilo_skupin,velikosti_skupin),]
tabela3<-tabela[order(stevilo_skupin,blocni_model,verjetnosti_povezav,velikosti_skupin),]
tabela4<-tabela[order(velikosti_skupin, blocni_model,verjetnosti_povezav,stevilo_skupin),]

############ Izvoz podatkov v excell ########################################
write.csv2(x=tabela1,file="tabela1.csv")
```

```
write.csv2(x=tabela2,file="tabela2.csv")
write.csv2(x=tabela3,file="tabela3.csv")
write.csv2(x=tabela4,file="tabela4.csv")

###### GRAFIČNI PRIKAZ aritmetičnih sredin ARI po posameznih spremenljivkah ##

#### GRAFIČNI PRIKAZ ARI - VERJETNOSTI POVEZAV ######################
win.graph()
mar.def<-c(5,4,4,1)+0.1
par(mar=mar.def+c(4,0,0,14),mfrow=c(1,1),xpd=TRUE)
matplot(tabela1[,1:8],type="o",xaxt="n",ylab="Povprečja", lty = 1:6, col = 1:6,pch = (18:25))
abline(v = at, col = "gray60")
legend(x = "right",legend = colnames(B), ncol=1,inset = -0.3,0,title = "Metode" ,lty = 1:6, lwd = 1, pch = (18:25), col = 1:6,
cex=0.8)
title(xlab="Nastavitve",line=7.5)
at<-as.vector(by(1:26,INDICES = tabela1$verjetnosti_povezav,FUN = max))
at<-at[1:(length(at)-1)]+0.5

axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.23)
at<-by(1:26,INDICES = tabela1$verjetnosti_povezav,FUN = mean)
names(at)
axis(side=1,at=at, line=3, labels= names(at),tick = FALSE, font=2)
axis(side=1,at=-0.8, line=3, labels= "Verjetnosti",tick = FALSE, font=2)

at<-as.vector(by(1:26,INDICES = paste(tabela1$verjetnosti_povezav,tabela1$blocni_model, sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.18)

at<-sort(by(1:26,INDICES = paste(tabela1$blocni_model,tabela1$verjetnosti_povezav, sep="|"),FUN = mean))
names(at)<-sapply(strsplit(names(at),split="|",fixed=TRUE),function(x)x[1])
axis(side=1,at=at, line=2, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=2, labels= "Bločni model",tick = FALSE)


at<-as.vector(by(1:26,INDICES = paste(tabela1$stevilo_skupin,tabela1$blocni_model, tabela1$verjetnosti_povezav,
sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.12)

at<-sort(by(1:26,INDICES = paste(tabela1$stevilo_skupin,tabela1$blocni_model, tabela1$verjetnosti_povezav, sep="|"),FUN =
mean))
names(at)<-sapply(strsplit(names(at),split="|",fixed=TRUE),function(x)x[1])
axis(side=1,at=at, line=1, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=1, labels= "Št. skupin",tick = FALSE)

at<-as.vector(by(1:26,INDICES = paste(tabela1$velikosti_skupin,tabela1$stevilo_skupin,tabela1$blocni_model,
tabela1$verjetnosti_povezav, sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.06)

at<-sort(by(1:26,INDICES = paste(tabela1$velikosti_skupin,tabela1$stevilo_skupin,tabela1$blocni_model,
tabela1$verjetnosti_povezav, sep="|"),FUN = mean))
names(at)<-substr(names(at),start = 1,stop = 1)
axis(side=1,at=at, line=0, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=0, labels= "Vel. skupin",tick = FALSE)

#### GRAFIČNI PRIKAZ ARI - BLOČNI MODEL ######################################
win.graph()
mar.def<-c(5,4,4,1)+0.1
par(mar=mar.def+c(4,0,0,14),mfrow=c(1,1),xpd=TRUE)
matplot(tabela2[,1:8],type="o",xaxt="n",ylab="Povprečja", lty = 1:6, col = 1:6, pch = (18:25))
legend(x = "right",legend = colnames(B), ncol=1,inset = -0.3,0,title = "Metode" ,lty = 1:6, lwd = 1, pch = (18:25), col = 1:6,
cex=0.8)
title(xlab="Nastavitve",line=7.5)
at<-as.vector(by(1:26,INDICES = tabela2$blocni_model,FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.23)

at<-by(1:26,INDICES = tabela2$blocni_model,FUN = mean)
names(at)
axis(side=1,at=at, line=3, labels= names(at),tick = FALSE,font=2)
axis(side=1,at=-0.8, line=3, labels= "Bločni model",tick = FALSE, font=2)

at<-as.vector(by(1:26,INDICES = paste(tabela2$verjetnosti_povezav,tabela2$blocni_model, sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.18)
```

```
at<-sort(by(1:26,INDICES = paste(tabela2$verjetnosti_povezav, tabela2$blocni_model,sep="|"),FUN = mean))
names(at)<-sapply(strsplit(names(at),split="|",fixed=TRUE),function(x)x[1])
axis(side=1,at=at, line=2, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=2, labels= "Verjetnosti",tick = FALSE)


at<-as.vector(by(1:26,INDICES = paste(tabela2$stevilo_skupin,tabela2$blocni_model, tabela2$verjetnosti_povezav,
sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.12)

at<-sort(by(1:26,INDICES = paste(tabela2$stevilo_skupin,tabela2$blocni_model, tabela2$verjetnosti_povezav, sep="|"),FUN =
mean))
names(at)<-sapply(strsplit(names(at),split="|",fixed=TRUE),function(x)x[1])
axis(side=1,at=at, line=1, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=1, labels= "Št. skupin",tick = FALSE)

at<-as.vector(by(1:26,INDICES = paste(tabela2$velikosti_skupin,tabela2$stevilo_skupin,tabela2$blocni_model,
tabela2$verjetnosti_povezav, sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.06)

at<-sort(by(1:26,INDICES = paste(tabela2$velikosti_skupin,tabela2$stevilo_skupin,tabela2$blocni_model,
tabela2$verjetnosti_povezav, sep="|"),FUN = mean))
names(at)<-substr(names(at),start = 1,stop = 1)
axis(side=1,at=at, line=0, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=0, labels= "Vel. skupin",tick = FALSE)


#### GRAFIČNI PRIKAZ ARI - ŠTEVILO SKUPIN ################################
win.graph()
mar.def<-c(5,4,4,1)+0.1
par(mar=mar.def+c(4,0,0,14),mfrow=c(1,1),xpd=TRUE)
matplot(tabela3[,1:8],type="o",xaxt="n",ylab="Povprečja", lty = 1:6, col = 1:6, pch = (18:25))
legend(x = "right",legend = colnames(B), ncol=1,inset = -0.3,0,title = "Metode" ,lty = 1:6, lwd = 1, pch = (18:25), col = 1:6,
cex=0.8)
title(xlab="Nastavitve",line=7.5)
at<-as.vector(by(1:26,INDICES = tabela3$stevilo_skupin,FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.23)

at<-by(1:26,INDICES = tabela3$stevilo_skupin,FUN = mean)
names(at)
axis(side=1,at=at, line=3, labels= names(at),tick = FALSE, font=2)
axis(side=1,at=-0.8, line=3, labels= "Št. skupin",tick = FALSE, font=2)

at<-as.vector(by(1:26,INDICES = paste(tabela3$stevilo_skupin,tabela3$blocni_model, sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.18)

at<-sort(by(1:26,INDICES = paste(tabela3$blocni_model, tabela3$stevilo_skupin,sep="|"),FUN = mean))
names(at)<-sapply(strsplit(names(at),split="|",fixed=TRUE),function(x)x[1])
axis(side=1,at=at, line=2, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=2, labels= "Bločni model",tick = FALSE)


at<-as.vector(by(1:26,INDICES = paste(tabela3$verjetnosti_povezav,tabela3$stevilo_skupin,tabela3$blocni_model,
sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.12)

at<-sort(by(1:26,INDICES = paste( tabela3$verjetnosti_povezav,tabela3$stevilo_skupin,tabela3$blocni_model, sep="|"),FUN =
mean))
names(at)<-sapply(strsplit(names(at),split="|",fixed=TRUE),function(x)x[1])
axis(side=1,at=at, line=1, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=1, labels= "Verjetnosti",tick = FALSE)

at<-as.vector(by(1:26,INDICES = paste(tabela3$velikosti_skupin,tabela3$stevilo_skupin,tabela3$blocni_model,
tabela3$verjetnosti_povezav, sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.06)

at<-sort(by(1:26,INDICES = paste(tabela3$velikosti_skupin,tabela3$stevilo_skupin,tabela3$blocni_model,
tabela3$verjetnosti_povezav, sep="|"),FUN = mean))
names(at)<-substr(names(at),start = 1,stop = 1)
```

```
axis(side=1,at=at, line=0, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=0, labels= "Vel. skupin",tick = FALSE)

#### GRAFIČNI PRIKAZ ARI - VELIKOSTI SKUPIN ##############################
win.graph()
mar.def<-c(5,4,4,1)+0.1
par(mar=mar.def+c(4,0,0,14),mfrow=c(1,1),xpd=TRUE)
matplot(tabela4[,1:8],type="o",xaxt="n",ylab="Povprečja", lty = 1:6, col = 1:6, pch = (18:25))
legend(x = "right",legend = colnames(B), ncol=1,inset = -0.3,0,title = "Metode" ,lty = 1:6, lwd = 1, pch = (18:25), col = 1:6,
cex=0.8)
title(xlab="Nastavitve",line=7.5)
at<-as.vector(by(1:26,INDICES = tabela4$velikosti_skupin,FUN = max))

at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.23)

at<-by(1:26,INDICES = tabela4$velikosti_skupin,FUN = mean)
names(at)

axis(side=1,at=at, line=3, labels= names(at),tick = FALSE, font=2)
axis(side=1,at=-0.8, line=3, labels= "Vel. skupin",tick = FALSE, font=2)

at<-as.vector(by(1:26,INDICES = paste(tabela4$velikosti_skupin,tabela4$blocni_model, sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.18)

at<-sort(by(1:26,INDICES = paste(tabela4$blocni_model, tabela4$velikosti_skupin,sep="|"),FUN = mean))
names(at)<-sapply(strsplit(names(at),split="|",fixed=TRUE),function(x)x[1])
axis(side=1,at=at, line=2, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=2, labels= "Bločni model",tick = FALSE)


at<-as.vector(by(1:26,INDICES = paste(tabela4$verjetnosti_povezav,tabela4$stevilo_skupin,tabela4$blocni_model,
sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.12)

at<-sort(by(1:26,INDICES = paste( tabela4$verjetnosti_povezav,tabela4$stevilo_skupin,tabela4$blocni_model, sep="|"),FUN =
mean))
names(at)<-sapply(strsplit(names(at),split="|",fixed=TRUE),function(x)x[1])
axis(side=1,at=at, line=1, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=1, labels= "Verjetnosti",tick = FALSE)

at<-as.vector(by(1:26,INDICES = paste(tabela4$stevilo_skupin,tabela4$velikosti_skupin,tabela4$blocni_model,
tabela4$verjetnosti_povezav, sep="|"),FUN = max))
at<-at[1:(length(at)-1)]+0.5
axis(side=1,at=at, line=0, labels= rep("",length(at)),tick = TRUE,tck=-0.06)

at<-sort(by(1:26,INDICES = paste(tabela4$stevilo_skupin,tabela4$velikosti_skupin,tabela4$blocni_model,
tabela4$verjetnosti_povezav, sep="|"),FUN = mean))
names(at)<-substr(names(at),start = 1,stop = 1)
axis(side=1,at=at, line=0, labels= names(at),tick = FALSE)
axis(side=1,at=-0.8, line=0, labels= "Število skupin",tick = FALSE)

##############################################################################
####### VEČFAKTORSKA ANALIZA VARIANCE

B<-rbind (a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,r,s,t,u,v,z,x,y,x1,y1)
stevilo_skupin<-factor(c(1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2))## 1- dve skupini; 2- 3 skupine
blocni_model<-factor(c(1,1,2,2,3,3,1,1,2,2,3,3,6,6,4,4,5,5,4,4,5,5,7,7,8,8))## 1- 1.model itd.
velikosti_skupin<-factor(c(1,1,1,1,1,1,2,2,2,2,2,2,1,2,1,1,1,1,2,2,2,2,1,2,1,2))## 1- enake velikosti, 2- različne velikosti
verjetnosti_povezav<-factor(c(1,2,1,2,1,2,1,2,1,2,1,2,3,3,1,2,1,2,1,2,1,2,3,3,3,3))## 1- 0.5, 2-0.3, 3- 0.5 in 0.3

summary(B)
summary(stevilo_skupin)
summary(blocni_model)
summary(velikosti_skupin)
summary(verjetnosti_povezav)

###### ANALIZA VARIANCE ##################################################

sapply(paste("res_",1:26,sep=""),function(x)nrow(get(x)))
stevilo_skupin<-factor(c(1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2),labels=c(2,3))## 1- dve skupini; 2- 3 skupine
blocni_model<-factor(c(1,1,2,2,3,3,1,1,2,2,3,3,6,6,4,4,5,5,4,4,5,5,7,7,8,8))## 1- 1.model itd.
blocni_model[blocni_model==4]<-3
blocni_model[blocni_model==6]<-1
```

```
blocni_model[blocni_model==7]<-3
blocni_model[blocni_model==8]<-1
blocni_model[blocni_model==5]<-1
blocni_model<-factor(blocni_model)
levels(blocni_model)<-c("CP","1P","Hom")

velikosti_skupin<-factor(c(1,1,1,1,1,1,2,2,2,2,2,2,1,2,1,1,1,1,2,2,2,2,1,2,1,2), labels=c("enake","različne"))## 1- enake velikosti,
2- različne velikosti
verjetnosti_povezav<-factor(c(1,2,1,2,1,2,1,2,1,2,1,2,3,3,1,2,1,2,1,2,1,2,3,3,3,3),labels=c(0.5, 0.3,"0.5 in 0.3"))

tabelaNastavitev<-
data.frame(sim=factor(1:26),stevilo_skupin=stevilo_skupin,velikosti_skupin=velikosti_skupin,blocni_model=blocni_model,verjetn
osti_povezav=verjetnosti_povezav)

mainRes<-NULL
for(i in 1:26){
  tmp<-cbind(get(paste("res_",i,sep="")),tabelaNastavitev[i,, drop=FALSE])
  mainRes<-rbind(mainRes,tmp)
}

mainResLong<-reshape(mainRes,direction =
"long",idvar="ponovitev",varying=list(colnames(mainRes)[1:8]),times=colnames(mainRes)[1:8], timevar="Metoda")
names(mainResLong)[7]<-"ARI"
mainRes[101,]
mainResLong[(0:7)*26*100+101,]

surovipodatki<-data.frame(mainResLong)
write.csv2(x=surovipodatki,file="surovipodatki.csv")
aov(ARI~stevilo_skupin*velikosti_skupin*blocni_model*verjetnosti_povezav*Metoda,data=mainResLong)
mod<-aov(ARI~blocni_model*stevilo_skupin*velikosti_skupin*verjetnosti_povezav*Metoda,data=mainResLong)
library(car)
Anova(mod,type="II")

############ Izvoz podatkov v excell ##########################################
tabela_anova<-data.frame(Anova(mod,type="II"))
write.csv2(x=tabela_anova,file="anova.csv")
```

Priloga B: Aritmetična sredina in standardni odklon za ARI

Tabela B.1: Aritmetična sredina in standardni odklon za ARI posameznih metod

	Nastavitve	binStr		binStrU		binReg		hom		homOm		sedist		mixVar		mixBay	
	k | bm | c | p	AS	SD	AS	SD	AS	SD	AS	SD	AS	SD	AS	SD	AS	SD	AS	SD
1	2 | BM_1 | 1 | 0.5	0.243	0.087	0.901	0.066	0.002	0.026	0.999	0.007	1.000	0.000	0.981	0.040	1.000	0.000	1.000	0.000
2	2 | BM_1 | 1 | 0.3	0.017	0.030	0.647	0.108	0.001	0.027	0.850	0.119	0.730	0.178	0.539	0.194	0.891	0.095	0.891	0.098
3	2 | BM_2 | 1 | 0.5	-0.001	0.009	0.689	0.000	-0.001	0.024	1.000	0.000	0.998	0.011	0.979	0.036	1.000	0.000	1.000	0.000
4	2 | BM_2 | 1 | 0.3	0.001	0.011	0.895	0.083	0.001	0.017	0.890	0.087	0.888	0.093	0.500	0.168	0.886	0.089	0.886	0.089
5	2 | BM_3 | 1 | 0.5	0.424	0.096	1.000	0.000	-0.003	0.019	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
6	2 | BM_3 | 1 | 0.3	0.003	0.021	0.988	0.027	0.001	0.026	0.987	0.030	0.935	0.168	0.820	0.113	0.986	0.030	0.986	0.030
7	2 | BM_1 | 2 | 0.5	0.440	0.101	0.999	0.007	0.002	0.026	0.999	0.007	1.000	0.000	0.997	0.016	1.000	0.000	1.000	0.000
8	2 | BM_1 | 2 | 0.3	0.108	0.060	0.937	0.061	0.006	0.039	0.940	0.066	0.700	0.185	0.773	0.136	0.946	0.066	0.946	0.066
9	2 | BM_2 | 2 | 0.5	0.037	0.000	1.000	0.000	-0.005	0.020	0.999	0.007	0.997	0.013	0.984	0.046	0.999	0.007	0.999	0.007
10	2 | BM_2 | 2 | 0.3	0.001	0.041	0.819	0.128	-0.006	0.020	0.895	0.090	0.897	0.087	0.559	0.185	0.891	0.095	0.891	0.092
11	2 | BM_3 | 2 | 0.5	0.233	0.090	1.000	0.000	0.002	0.029	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000
12	2 | BM_3 | 2 | 0.3	-0.017	0.039	0.978	0.046	0.004	0.028	0.977	0.047	0.760	0.349	0.841	0.133	0.982	0.038	0.983	0.036
13	2 | BM_6 | 1 | 0.5/0.3	0.363	0.100	0.560	0.065	0.002	0.033	0.966	0.050	0.966	0.043	0.837	0.121	0.974	0.044	0.974	0.044
14	2 | BM_6 | 2 | 0.5/0.3	0.233	0.102	0.884	0.065	-0.003	0.020	0.954	0.060	0.953	0.051	0.794	0.119	0.963	0.050	0.963	0.050
15	3 | BM_4 | 1 | 0.5	0.330	0.104	1.000	0.000	-0.005	0.021	1.000	0.000	1.000	0.000	0.989	0.024	1.000	0.000	1.000	0.000
16	3 | BM_4 | 1 | 0.3	0.026	0.036	0.932	0.066	-0.008	0.020	0.879	0.141	0.499	0.272	0.467	0.179	0.820	0.247	0.817	0.212
17	3 | BM_5 | 1 | 0.5	0.246	0.076	0.909	0.055	-0.004	0.026	0.983	0.029	0.992	0.020	0.876	0.085	0.991	0.021	0.991	0.021
18	3 | BM_5 | 1 | 0.3	0.028	0.029	0.587	0.113	-0.003	0.021	0.548	0.166	0.340	0.154	0.286	0.086	0.492	0.192	0.467	0.172
19	3 | BM_4 | 2 | 0.5	0.130	0.082	0.999	0.004	-0.001	0.026	0.905	0.210	0.757	0.270	0.875	0.221	0.640	0.204	0.767	0.198
20	3 | BM_4 | 2 | 0.3	-0.109	0.054	0.885	0.126	0.016	0.044	0.406	0.069	0.385	0.125	0.376	0.100	0.523	0.113	0.688	0.182
21	3 | BM_5 | 2 | 0.5	0.062	0.067	0.133	0.070	0.000	0.021	0.486	0.173	0.582	0.262	0.341	0.152	0.586	0.203	0.607	0.188
22	3 | BM_5 | 2 | 0.3	-0.021	0.066	0.064	0.056	0.000	0.021	0.110	0.093	0.082	0.137	0.080	0.102	0.189	0.158	0.211	0.196
23	3 | BM_7 | 1 | 0.5/0.3	0.255	0.067	0.788	0.086	0.005	0.025	0.841	0.097	0.692	0.132	0.641	0.131	0.644	0.206	0.644	0.181
24	3 | BM_7 | 2 | 0.5/0.3	0.140	0.083	0.829	0.201	0.006	0.000	0.447	0.020	0.476	0.085	0.484	0.062	0.554	0.112	0.703	0.181

25	3	BM_8	1	0.5/0.3	0.239	0.070	0.785	0.111	0.006	0.026	0.807	0.111	0.596	0.124	0.474	0.138	0.711	0.206	0.651	0.202
26	3	BM_8	2	0.5/0.3	0.126	0.075	0.524	0.164	0.034	0.072	0.766	0.230	0.886	0.147	0.536	0.193	0.787	0.240	0.817	0.215

Legenda: k – število skupin, bm – bločni model, c – velikost skupin (1 – enako število skupin, 2 – različno število skupin), p – verjetnosti po povezavah za polne bloke

Slika B.1: Grafični prikaz standardnega odklona ARI