Networks

Mihalis Galanakis

14/5/2022

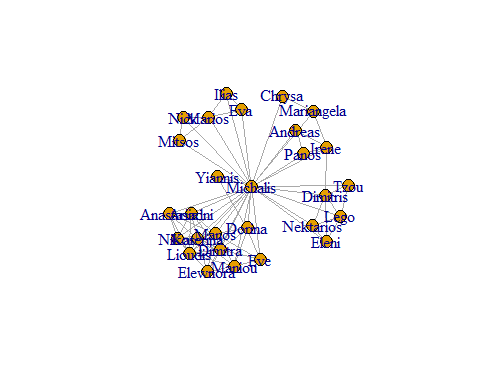
**Create my social network**

# install.packages('igraph')  
library(igraph)

m1 <- graph( ~'Lioudis'-'Ariadni'-'Michalis'-'Elewnora'-'Manos'-'Ariadni'-'Anastasia'-'Lioudis'-'Nikos'-'Michalis',  
'Nikos'-'Manos'-'Anastasia'-'Nikos', 'Nikos'-'Ariadni',  
'Michalis'-'Manos'-'Lioudis', 'Lioudis'-'Michalis'-'Anastasia', 'Elewnora'-'Nikos', 'Michalis'-'Katerina'-'Manos',   
'Ariadni'-'Katerina'-'Elewnora', 'Anastasia'-'Katerina'-'Lioudis', 'Nikos'-'Katerina'-'Manos',  
'Michalis'-'Dimitra'-'Manos', 'Katerina'-'Dimitra'-'Nikos', 'Ariadni'-'Dimitra'-'Elewnora', 'Lioudis'-'Dimitra',  
'Michalis'-'Maniou'-'Katerina', 'Lioudis'-'Maniou'-'Manos', 'Elewnora'-'Maniou',  
'Michalis'-'Dorina'-'Maniou', 'Dimitra'-'Dorina'-'Manos', 'Michalis'-'Eve'-'Dorina', 'Maniou'-'Eve'-'Dimitra',   
'Michalis'-'Eve'-'Manos', 'Michalis'-'Yiannis'-'Dorina',  
'Michalis'-'Marios'-'Ilias'-'Eva', 'Michalis'-'Eva'-'Marios', 'Michalis'-'Ilias',  
'Lego'-'Dimitris'-'Michalis'-'Nektarios', 'Dimitris'-'Nektarios'-'Lego'-'Michalis'-'Tzou','Dimitris'-'Tzou'-'Lego',  
'Marios'-'Mitsos'-'Michalis'-'Nick'-'Marios','Marios'-'Nick'-'Mitsos',  
'Mariangela'-'Michalis'-'Irene'-'Mariangela', 'Panos'-'Irene'-'Andreas'-'Panos'-'Michalis'-'Andreas',  
'Michalis'-'Chrysa'-'Mariangela', 'Michalis'-'Eleni'-'Lego','Nektarios'-'Eleni'-'Dimitris', 'Irene'-'Dimitris'  
)

**Plot my social network**

plot(m1,edge.arrow.size=1)



**Adjacency matrix**

m1[]

## 28 x 28 sparse Matrix of class "dgCMatrix"

## [[ suppressing 28 column names 'Lioudis', 'Ariadni', 'Michalis' ... ]]

##   
## Lioudis . 1 1 . 1 1 1 1 1 1 . . . . . . . . . . . . . . . . . .  
## Ariadni 1 . 1 . 1 1 1 1 1 . . . . . . . . . . . . . . . . . . .  
## Michalis 1 1 . 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## Elewnora . . 1 . 1 . 1 1 1 1 . . . . . . . . . . . . . . . . . .  
## Manos 1 1 1 1 . 1 1 1 1 1 1 1 . . . . . . . . . . . . . . . .  
## Anastasia 1 1 1 . 1 . 1 1 . . . . . . . . . . . . . . . . . . . .  
## Nikos 1 1 1 1 1 1 . 1 1 . . . . . . . . . . . . . . . . . . .  
## Katerina 1 1 1 1 1 1 1 . 1 1 . . . . . . . . . . . . . . . . . .  
## Dimitra 1 1 1 1 1 . 1 1 . . 1 1 . . . . . . . . . . . . . . . .  
## Maniou 1 . 1 1 1 . . 1 . . 1 1 . . . . . . . . . . . . . . . .  
## Dorina . . 1 . 1 . . . 1 1 . 1 1 . . . . . . . . . . . . . . .  
## Eve . . 1 . 1 . . . 1 1 1 . . . . . . . . . . . . . . . . .  
## Yiannis . . 1 . . . . . . . 1 . . . . . . . . . . . . . . . . .  
## Marios . . 1 . . . . . . . . . . . 1 1 . . . . 1 1 . . . . . .  
## Ilias . . 1 . . . . . . . . . . 1 . 1 . . . . . . . . . . . .  
## Eva . . 1 . . . . . . . . . . 1 1 . . . . . . . . . . . . .  
## Lego . . 1 . . . . . . . . . . . . . . 1 1 1 . . . . . . . 1  
## Dimitris . . 1 . . . . . . . . . . . . . 1 . 1 1 . . . 1 . . . 1  
## Nektarios . . 1 . . . . . . . . . . . . . 1 1 . . . . . . . . . 1  
## Tzou . . 1 . . . . . . . . . . . . . 1 1 . . . . . . . . . .  
## Mitsos . . 1 . . . . . . . . . . 1 . . . . . . . 1 . . . . . .  
## Nick . . 1 . . . . . . . . . . 1 . . . . . . 1 . . . . . . .  
## Mariangela . . 1 . . . . . . . . . . . . . . . . . . . . 1 . . 1 .  
## Irene . . 1 . . . . . . . . . . . . . . 1 . . . . 1 . 1 1 . .  
## Panos . . 1 . . . . . . . . . . . . . . . . . . . . 1 . 1 . .  
## Andreas . . 1 . . . . . . . . . . . . . . . . . . . . 1 1 . . .  
## Chrysa . . 1 . . . . . . . . . . . . . . . . . . . 1 . . . . .  
## Eleni . . 1 . . . . . . . . . . . . . 1 1 1 . . . . . . . . .

# The edges of the object  
E(m1)

## + 83/83 edges from c23acda (vertex names):  
## [1] Lioudis --Ariadni Lioudis --Michalis Lioudis --Manos   
## [4] Lioudis --Anastasia Lioudis --Nikos Lioudis --Katerina   
## [7] Lioudis --Dimitra Lioudis --Maniou Ariadni --Michalis   
## [10] Ariadni --Manos Ariadni --Anastasia Ariadni --Nikos   
## [13] Ariadni --Katerina Ariadni --Dimitra Michalis--Elewnora   
## [16] Michalis--Manos Michalis--Anastasia Michalis--Nikos   
## [19] Michalis--Katerina Michalis--Dimitra Michalis--Maniou   
## [22] Michalis--Dorina Michalis--Eve Michalis--Yiannis   
## [25] Michalis--Marios Michalis--Ilias Michalis--Eva   
## [28] Michalis--Lego Michalis--Dimitris Michalis--Nektarios   
## + ... omitted several edges

# The vertices of the object  
V(m1)

## + 28/28 vertices, named, from c23acda:  
## [1] Lioudis Ariadni Michalis Elewnora Manos Anastasia   
## [7] Nikos Katerina Dimitra Maniou Dorina Eve   
## [13] Yiannis Marios Ilias Eva Lego Dimitris   
## [19] Nektarios Tzou Mitsos Nick Mariangela Irene   
## [25] Panos Andreas Chrysa Eleni

**Add attributes to the network**

V(m1)$name

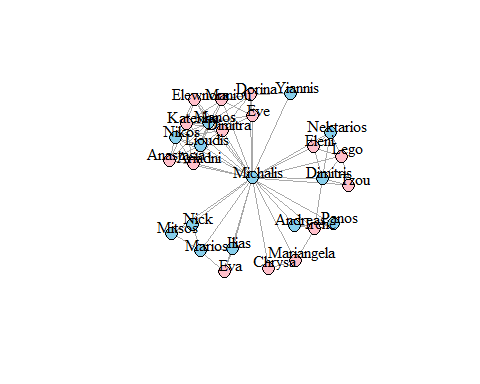
## [1] "Lioudis" "Ariadni" "Michalis" "Elewnora" "Manos"   
## [6] "Anastasia" "Nikos" "Katerina" "Dimitra" "Maniou"   
## [11] "Dorina" "Eve" "Yiannis" "Marios" "Ilias"   
## [16] "Eva" "Lego" "Dimitris" "Nektarios" "Tzou"   
## [21] "Mitsos" "Nick" "Mariangela" "Irene" "Panos"   
## [26] "Andreas" "Chrysa" "Eleni"

V(m1)$gender <- c("male", "female", "male", "female", "male", "female", "male",'female','female','female','female','female',  
'male','male','male','female','female','male','male','female','male','male','female','female','male','male','female','female')  
vertex\_attr(m1)

## $name  
## [1] "Lioudis" "Ariadni" "Michalis" "Elewnora" "Manos"   
## [6] "Anastasia" "Nikos" "Katerina" "Dimitra" "Maniou"   
## [11] "Dorina" "Eve" "Yiannis" "Marios" "Ilias"   
## [16] "Eva" "Lego" "Dimitris" "Nektarios" "Tzou"   
## [21] "Mitsos" "Nick" "Mariangela" "Irene" "Panos"   
## [26] "Andreas" "Chrysa" "Eleni"   
##   
## $gender  
## [1] "male" "female" "male" "female" "male" "female" "male" "female"  
## [9] "female" "female" "female" "female" "male" "male" "male" "female"  
## [17] "female" "male" "male" "female" "male" "male" "female" "female"  
## [25] "male" "male" "female" "female"

**Visualization**

plot(m1, edge.arrow.size=.5, vertex.label.color="black", vertex.label.dist=1.5,  
 vertex.color=c( "pink", "skyblue")[1+(V(m1)$gender=="male")] )



**Network descriptive statistics**

vcount(m1)

## [1] 28

# Number of nodes  
  
ecount(m1)

## [1] 83

# Number of edges  
  
# Density: The proportion of present edges from all possible edges in thenetwork.  
edge\_density(m1)

## [1] 0.2195767

# Note that it's the same as:  
ecount(m1)/(vcount(m1)\*(vcount(m1)-1)/2)

## [1] 0.2195767

reciprocity(m1)

## [1] 1

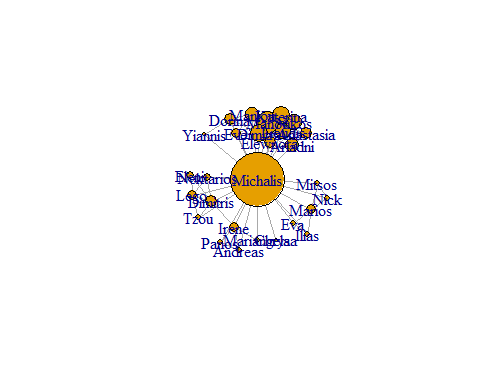
# Always 1 for undirected graphs!  
  
  
# Transitivity: ratio of triangles (direction disregarded) to connected triples  
transitivity(m1, type="global")

## [1] 0.4705882

# Diameter: A network diameter is the longest geodesic distance (length of the shortest path between two nodes) in the network.  
diameter(m1)

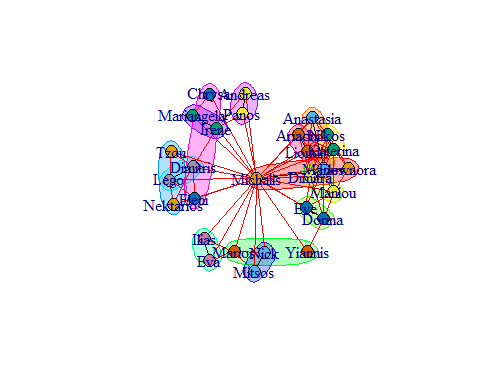
## [1] 2

# Node degree: The function degree() has a mode of in for in-degree, out for out-degree, and all or total for total degree.  
deg <- degree(m1, mode="all")  
plot(m1, vertex.size=deg\*3)



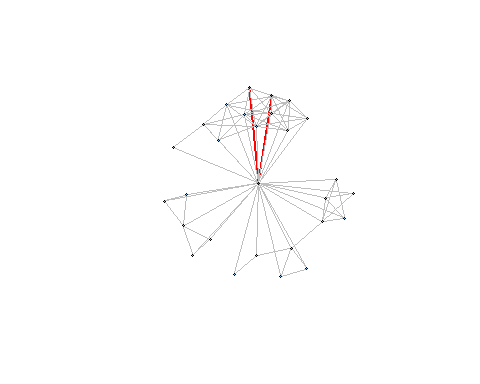
**Community detection**

# Fast hierarchical agglomeration algorithm for finding community structure (Clauset, Newman, and Moore, 2004)  
library(igraph)  
m <- make\_graph(V(m1)$name)  
fc <- cluster\_fast\_greedy(as.undirected(m))  
plot(fc,m1)

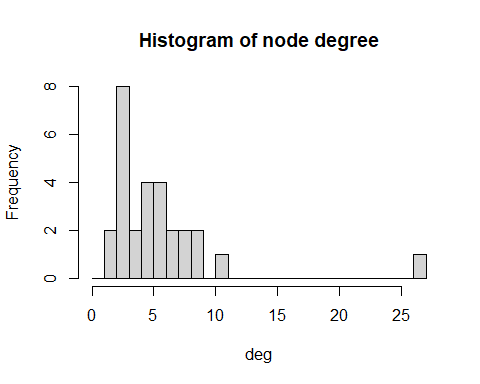


**Visualizations**

d = get.diameter(m1)  
E(m1)$color = "grey"  
E(m1)$width = 1  
E(m1, path=d)$color = "red"  
E(m1, path=d)$width = 2  
V(m1)$color = "SkyBlue2"  
V(m1)[d]$color = "red"  
coords = layout.fruchterman.reingold(m1)  
plot(m1, layout=coords, vertex.label = NA, vertex.size=3)



hist(deg, breaks=1:vcount(m1)-1, main="Histogram of node degree")



**I’m the guy with the most connections obviously!**

**Convert the igraph object to a network one**

# install.packages('intergraph')  
library(network)

library(intergraph)

net.m1 <- asNetwork(m1)

**Fitting the Erdos-Renyi model**

# install.packages('ergm')  
library(ergm)

# The ergm package fits the model using Monte Carlo maximum likelihood estimation (MCMLE).It implements maximum likelihood estimates of ERGMs to be calculated using Markov Chain Monte Carlo  
  
model1 <- ergm(net.m1 ~ edges)

## Starting maximum pseudolikelihood estimation (MPLE):

## Evaluating the predictor and response matrix.

## Maximizing the pseudolikelihood.

## Finished MPLE.

## Stopping at the initial estimate.

## Evaluating log-likelihood at the estimate.

model1

##   
## Call:  
## ergm(formula = net.m1 ~ edges)  
##   
## Maximum Likelihood Coefficients:  
## edges   
## -1.268

summary(model1)

## Call:  
## ergm(formula = net.m1 ~ edges)  
##   
## Maximum Likelihood Results:  
##   
## Estimate Std. Error MCMC % z value Pr(>|z|)   
## edges -1.2681 0.1242 0 -10.21 <1e-04 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Null Deviance: 524.0 on 378 degrees of freedom  
## Residual Deviance: 397.9 on 377 degrees of freedom  
##   
## AIC: 399.9 BIC: 403.9 (Smaller is better. MC Std. Err. = 0)

**Metrics of Gof for model1**

control.ergm(seed = 2345)

## Control parameter list generated by 'control.ergm' or equivalent. Non-empty parameters:  
## drop: TRUE  
## main.method: "MCMLE"  
## force.main: FALSE  
## main.hessian: TRUE  
## MPLE.samplesize: 2147483647L  
## init.MPLE.samplesize: function (d, e) max(sqrt(d), e, 40) \* 8  
## MPLE.type: "glm"  
## MPLE.maxit: 10000  
## MPLE.nonvar: "warning"  
## MPLE.nonident: "warning"  
## MPLE.nonident.tol: 1e-10  
## MPLE.constraints.ignore: FALSE  
## MCMC.prop: ~sparse  
## MCMC.prop.weights: "default"  
## MCMC.effectiveSize.damp: 10  
## MCMC.effectiveSize.maxruns: 16  
## MCMC.effectiveSize.burnin.pval: 0.2  
## MCMC.effectiveSize.burnin.min: 0.05  
## MCMC.effectiveSize.burnin.max: 0.5  
## MCMC.effectiveSize.burnin.nmin: 16  
## MCMC.effectiveSize.burnin.nmax: 128  
## MCMC.effectiveSize.burnin.PC: FALSE  
## MCMC.effectiveSize.burnin.scl: 32  
## MCMC.return.stats: TRUE  
## MCMC.runtime.traceplot: FALSE  
## MCMC.maxedges: Inf  
## MCMC.addto.se: TRUE  
## SAN.maxit: 4  
## SAN.nsteps.times: 8  
## SAN: Control parameter list generated by 'control.san' or equivalent. Non-empty parameters:  
## SAN.maxit: 4  
## SAN.tau: 1  
## SAN.invcov.diag: FALSE  
## SAN.nsteps.alloc: function (nsim) 2^seq\_len(nsim)  
## SAN.nsteps: 131072  
## SAN.samplesize: 1024  
## SAN.prop: ~sparse  
## SAN.prop.weights: "default"  
## SAN.ignore.finite.offsets: TRUE  
## parallel: 0  
## parallel.version.check: TRUE  
## parallel.inherit.MT: FALSE  
## MCMLE.termination: "confidence"  
## MCMLE.maxit: 60  
## MCMLE.conv.min.pval: 0.5  
## MCMLE.confidence: 0.99  
## MCMLE.confidence.boost: 2  
## MCMLE.confidence.boost.threshold: 1  
## MCMLE.confidence.boost.lag: 4  
## MCMLE.NR.maxit: 100  
## MCMLE.NR.reltol: 1.49011611938477e-08  
## obs.MCMC.mul: 0.25  
## obs.MCMC.samplesize.mul: 0.5  
## obs.MCMC.interval.mul: 0.5  
## obs.MCMC.burnin.mul: 0.5  
## obs.MCMC.prop: ~sparse  
## obs.MCMC.prop.weights: "default"  
## obs.MCMC.impute.min\_informative: function (nw) network.size(nw)/4  
## obs.MCMC.impute.default\_density: function (nw) 2/network.size(nw)  
## MCMLE.min.depfac: 2  
## MCMLE.sampsize.boost.pow: 0.5  
## MCMLE.MCMC.precision: 0.1  
## MCMLE.MCMC.max.ESS.frac: 0.1  
## MCMLE.metric: "lognormal"  
## MCMLE.method: "BFGS"  
## MCMLE.dampening: FALSE  
## MCMLE.dampening.min.ess: 20  
## MCMLE.dampening.level: 0.1  
## MCMLE.steplength.margin: 0.05  
## MCMLE.steplength: 1  
## MCMLE.steplength.parallel: "observational"  
## MCMLE.sequential: TRUE  
## MCMLE.density.guard.min: 10000  
## MCMLE.density.guard: 20.0855369231877  
## MCMLE.effectiveSize: 64  
## obs.MCMLE.effectiveSize: 16  
## MCMLE.interval: 1024  
## MCMLE.burnin: 16384  
## MCMLE.samplesize.per\_theta: 32  
## MCMLE.samplesize.min: 256  
## obs.MCMLE.samplesize.per\_theta: 16  
## obs.MCMLE.samplesize.min: 256  
## obs.MCMLE.interval: 512  
## obs.MCMLE.burnin: 8192  
## MCMLE.steplength.solver: c("glpk", "lpsolve")  
## MCMLE.last.boost: 4  
## MCMLE.steplength.esteq: TRUE  
## MCMLE.steplength.miss.sample: function (x1) ceiling(sqrt(ncol(rbind(x1))))  
## MCMLE.steplength.min: 1e-04  
## MCMLE.effectiveSize.interval\_drop: 2  
## MCMLE.nonvar: "message"  
## MCMLE.nonident: "warning"  
## MCMLE.nonident.tol: 1e-10  
## SA.nsubphases: 4  
## SA.interval: 1024  
## SA.burnin: 16384  
## SA.samplesize: 1024  
## RM.phase1n\_base: 7  
## RM.phase2n\_base: 100  
## RM.phase2sub: 7  
## RM.init\_gain: 0.5  
## RM.phase3n: 500  
## RM.interval: 1024  
## RM.burnin: 16384  
## RM.samplesize: 1024  
## Step.maxit: 50  
## Step.gridsize: 100  
## Step.interval: 1024  
## Step.burnin: 16384  
## Step.samplesize: 1024  
## CD.samplesize.per\_theta: 128  
## obs.CD.samplesize.per\_theta: 128  
## CD.nsteps: 8  
## CD.multiplicity: 1  
## CD.nsteps.obs: 128  
## CD.multiplicity.obs: 1  
## CD.maxit: 60  
## CD.conv.min.pval: 0.5  
## CD.NR.maxit: 100  
## CD.NR.reltol: 1.49011611938477e-08  
## CD.metric: "naive"  
## CD.method: "BFGS"  
## CD.dampening: FALSE  
## CD.dampening.min.ess: 20  
## CD.dampening.level: 0.1  
## CD.steplength.margin: 0.5  
## CD.steplength: 1  
## CD.adaptive.epsilon: 0.01  
## CD.steplength.esteq: TRUE  
## CD.steplength.miss.sample: function (x1) ceiling(sqrt(ncol(rbind(x1))))  
## CD.steplength.min: 1e-04  
## CD.steplength.parallel: "observational"  
## CD.steplength.solver: c("glpk", "lpsolve")  
## loglik: Control parameter list generated by 'control.logLik.ergm' or equivalent. Non-empty parameters:  
## bridge.nsteps: 16  
## bridge.bidirectional: TRUE  
## parallel.version.check: TRUE  
## parallel.inherit.MT: FALSE  
## seed: 2345  
## parallel: 0  
## parallel.version.check: TRUE  
## parallel.inherit.MT: FALSE

m1gof <- gof(model1, GOF = ~ distance + espartners + triadcensus,  
 verbose = TRUE, interval = 5e+4)

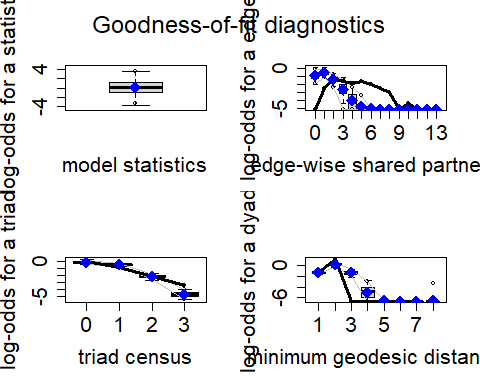
## Starting GOF for the given ERGM formula.

## Calculating observed network statistics.

## Starting simulations.

## Starting MCMC iterations to generate 100 networks

par(mfrow = c(2,2))  
plot(m1gof, cex.lab=1.6, cex.axis=1.6, plotlogodds = TRUE)



# It's visible that the fit is really bad in this model!

**Fit latent cluster random effect models**

# install.packages('latentnet')  
library(latentnet)

# The main idea is: Each node has a latent-unobserved position in a d-dimensional Euclidean space (e.g. d = 2). Each node is associated with one (unobserved) cluster. The model can also take into account covariates.  
# Setting up the model with G=2 clusters  
model.fit <- ergmm(net.m1 ~ euclidean(d = 2, G = 2), verbose = TRUE)

## Generating initial values for MCMC:  
## Computing geodesic distances... Finished.  
## Computing MDS locations... Finished.  
## Computing other initial values... Finished.  
## Finding the conditional posterior mode... Finished.  
## Burning in... Finished.  
## Starting sampling run... Finished.  
## Post-processing the MCMC output:  
## Performing label-switching... Finished.  
## Fitting the MKL locations... Finished.  
## Fitting MBC conditional on MKL locations... Finished.  
## Performing Procrustes transformation... Finished.

summary(model.fit)

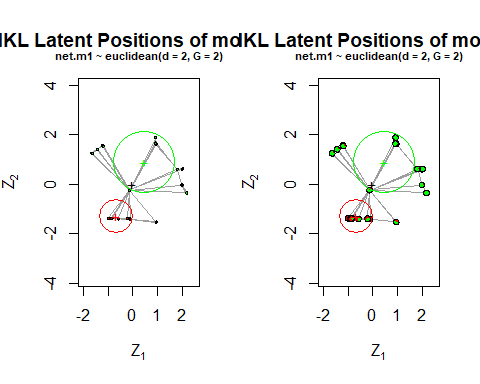
## NOTE: It is not certain whether it is appropriate to use latentnet's BIC to select latent space dimension, whether or not to include actor-specific random effects, and to compare clustered models with the unclustered model.

##   
## ==========================  
## Summary of model fit  
## ==========================  
##   
## Formula: net.m1 ~ euclidean(d = 2, G = 2)  
## Attribute: edges  
## Model: Bernoulli   
## MCMC sample of size 4000, draws are 10 iterations apart, after burnin of 10000 iterations.  
## Covariate coefficients posterior means:  
## Estimate 2.5% 97.5% 2\*min(Pr(>0),Pr(<0))   
## (Intercept) 2.3931 1.5291 3.2717 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Overall BIC: 366.9708   
## Likelihood BIC: 244.9889   
## Latent space/clustering BIC: 121.9819   
##   
## Covariate coefficients MKL:  
## Estimate  
## (Intercept) 0.6228339

# Posterior membership probabilities  
attr(model.fit$sample, "Q")

## [,1] [,2]  
## [1,] 0.80918193 0.1908181  
## [2,] 0.78939748 0.2106025  
## [3,] 0.30137411 0.6986259  
## [4,] 0.76535110 0.2346489  
## [5,] 0.82121352 0.1787865  
## [6,] 0.76128185 0.2387182  
## [7,] 0.80217593 0.1978241  
## [8,] 0.81541997 0.1845800  
## [9,] 0.80882105 0.1911790  
## [10,] 0.74160596 0.2583940  
## [11,] 0.59948891 0.4005111  
## [12,] 0.62909482 0.3709052  
## [13,] 0.34325396 0.6567460  
## [14,] 0.16217016 0.8378298  
## [15,] 0.16402877 0.8359712  
## [16,] 0.16270905 0.8372909  
## [17,] 0.07203349 0.9279665  
## [18,] 0.07087062 0.9291294  
## [19,] 0.07596861 0.9240314  
## [20,] 0.08322871 0.9167713  
## [21,] 0.16591259 0.8340874  
## [22,] 0.16308947 0.8369105  
## [23,] 0.11183876 0.8881612  
## [24,] 0.09159381 0.9084062  
## [25,] 0.10055324 0.8994468  
## [26,] 0.09808502 0.9019150  
## [27,] 0.12262172 0.8773783  
## [28,] 0.07616451 0.9238355

par(mfrow = c(1,2))  
plot(model.fit)  
plot(model.fit, pie = TRUE, vertex.cex = 2.5)



# Setting up the model with G=3 clusters  
model.fit.2 <- ergmm(net.m1 ~ euclidean(d = 2, G = 3), verbose = TRUE)

## Generating initial values for MCMC:  
## Computing geodesic distances... Finished.  
## Computing MDS locations... Finished.  
## Computing other initial values... Finished.  
## Finding the conditional posterior mode... Finished.  
## Burning in... Finished.  
## Starting sampling run... Finished.  
## Post-processing the MCMC output:  
## Performing label-switching... Finished.  
## Fitting the MKL locations... Finished.  
## Fitting MBC conditional on MKL locations... Finished.  
## Performing Procrustes transformation... Finished.

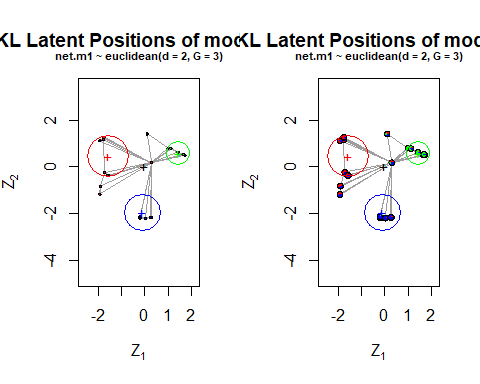
summary(model.fit.2)

##   
## ==========================  
## Summary of model fit  
## ==========================  
##   
## Formula: net.m1 ~ euclidean(d = 2, G = 3)  
## Attribute: edges  
## Model: Bernoulli   
## MCMC sample of size 4000, draws are 10 iterations apart, after burnin of 10000 iterations.  
## Covariate coefficients posterior means:  
## Estimate 2.5% 97.5% 2\*min(Pr(>0),Pr(<0))   
## (Intercept) 2.2051 1.4070 3.1363 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Overall BIC: 377.993   
## Likelihood BIC: 245.2305   
## Latent space/clustering BIC: 132.7625   
##   
## Covariate coefficients MKL:  
## Estimate  
## (Intercept) 0.6451395

# Posterior membership probabilities  
attr(model.fit.2$sample, "Q")

## [,1] [,2] [,3]  
## [1,] 0.03292406 0.92708985 0.03998609  
## [2,] 0.03986335 0.91355746 0.04657919  
## [3,] 0.36999222 0.31174604 0.31826175  
## [4,] 0.05952097 0.88076718 0.05971185  
## [5,] 0.02949643 0.93408665 0.03641693  
## [6,] 0.05094348 0.89108401 0.05797251  
## [7,] 0.03369579 0.92765193 0.03865228  
## [8,] 0.02920154 0.93436994 0.03642851  
## [9,] 0.03513163 0.92409118 0.04077719  
## [10,] 0.08237796 0.83119661 0.08642543  
## [11,] 0.17293302 0.64001311 0.18705388  
## [12,] 0.14627209 0.69483370 0.15889421  
## [13,] 0.35274985 0.26655722 0.38069294  
## [14,] 0.08278296 0.01850639 0.89871065  
## [15,] 0.10092490 0.02516347 0.87391163  
## [16,] 0.09814215 0.02446669 0.87739117  
## [17,] 0.84668838 0.01567225 0.13763937  
## [18,] 0.85143082 0.01444925 0.13411993  
## [19,] 0.84046611 0.01711204 0.14242185  
## [20,] 0.82440647 0.02111749 0.15447604  
## [21,] 0.10383938 0.02264265 0.87351797  
## [22,] 0.10497089 0.02414969 0.87087942  
## [23,] 0.61801735 0.02599867 0.35598398  
## [24,] 0.69252609 0.01589914 0.29157477  
## [25,] 0.65507836 0.02118285 0.32373878  
## [26,] 0.65624461 0.02039742 0.32335798  
## [27,] 0.58005991 0.03291343 0.38702666  
## [28,] 0.84003932 0.01821103 0.14174965

par(mfrow = c(1,2))  
plot(model.fit.2)  
plot(model.fit.2, pie = TRUE, vertex.cex = 2.5)



# Setting up the model with G=4 clusters  
model.fit.3 <- ergmm(net.m1 ~ euclidean(d = 2, G = 4), verbose = TRUE)

## Generating initial values for MCMC:  
## Computing geodesic distances... Finished.  
## Computing MDS locations... Finished.  
## Computing other initial values... Finished.  
## Finding the conditional posterior mode... Finished.  
## Burning in... Finished.  
## Starting sampling run... Finished.  
## Post-processing the MCMC output:  
## Performing label-switching... Finished.  
## Fitting the MKL locations... Finished.  
## Fitting MBC conditional on MKL locations... Finished.  
## Performing Procrustes transformation... Finished.

summary(model.fit.3)

##   
## ==========================  
## Summary of model fit  
## ==========================  
##   
## Formula: net.m1 ~ euclidean(d = 2, G = 4)  
## Attribute: edges  
## Model: Bernoulli   
## MCMC sample of size 4000, draws are 10 iterations apart, after burnin of 10000 iterations.  
## Covariate coefficients posterior means:  
## Estimate 2.5% 97.5% 2\*min(Pr(>0),Pr(<0))   
## (Intercept) 2.2207 1.4405 3.0964 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Overall BIC: 345.1997   
## Likelihood BIC: 243.5468   
## Latent space/clustering BIC: 101.6529   
##   
## Covariate coefficients MKL:  
## Estimate  
## (Intercept) 0.6837366

# Posterior membership probabilities  
attr(model.fit.3$sample, "Q")

## [,1] [,2] [,3] [,4]  
## [1,] 0.01746372 0.930158426 0.04001245 0.01236540  
## [2,] 0.02291983 0.915206202 0.04598618 0.01588779  
## [3,] 0.18557272 0.388937803 0.23158925 0.19390023  
## [4,] 0.03504061 0.863429181 0.07804231 0.02348791  
## [5,] 0.01836093 0.923822209 0.04663543 0.01118143  
## [6,] 0.03017744 0.890129014 0.05936566 0.02032788  
## [7,] 0.01913395 0.925012436 0.04188196 0.01397165  
## [8,] 0.01595175 0.935009396 0.03874537 0.01029348  
## [9,] 0.02071881 0.910976634 0.05399500 0.01430956  
## [10,] 0.06203505 0.786365492 0.11977894 0.03182052  
## [11,] 0.13852173 0.568719702 0.22018277 0.07257581  
## [12,] 0.11596665 0.628433336 0.19330585 0.06229417  
## [13,] 0.28393822 0.219815030 0.32220297 0.17404378  
## [14,] 0.91892546 0.003337808 0.04322296 0.03451377  
## [15,] 0.88671594 0.006981061 0.06252512 0.04377788  
## [16,] 0.88418400 0.006496773 0.06299789 0.04632133  
## [17,] 0.04950807 0.005135898 0.05631009 0.88904594  
## [18,] 0.04894698 0.004172561 0.06703463 0.87984584  
## [19,] 0.05239539 0.005391222 0.06333880 0.87887458  
## [20,] 0.07136849 0.008338600 0.08812589 0.83216702  
## [21,] 0.88575623 0.008186979 0.05802679 0.04803000  
## [22,] 0.88912299 0.006768618 0.05757006 0.04653833  
## [23,] 0.15118245 0.009127898 0.60693423 0.23275542  
## [24,] 0.12016258 0.005377731 0.59089146 0.28356823  
## [25,] 0.13752959 0.006022671 0.60770298 0.24874476  
## [26,] 0.13595213 0.006761509 0.60030303 0.25698333  
## [27,] 0.17226975 0.013171864 0.58316362 0.23139476  
## [28,] 0.05374695 0.005417034 0.06516656 0.87566946

par(mfrow = c(1,2))  
plot(model.fit.3)  
plot(model.fit.3, pie = TRUE, vertex.cex = 2.5)

