**[For RESEARCH Section]**

**Evidence for extensive social networks as risk-mitigation strategies on south-west Madagascar**

Dylan S. Davis1,2\*, Tanambelo Rasolondrainy3,4, George Manahira4, Sean Hixon5, Vanillah Andriankaja4, Laurance Hubertine4, Ricky Justome4, François Lahiniriko4, Harson Léonce4, Razafimagnefa Roi4, Faralahy Victorian4, Marius Brenah Jean Clovis4, Vavisoa Voahirana4, Tahirisoa Lorine Carina4, Augustin Jean Yve4, Zafy Maharesy Chrisostome4†, Barthélémy Manjakahery3† & Kristina Douglass1,2,4

1 Columbia Climate School, Columbia University, New York, USA

2 Division of Biology and Paleoenvironment, Lamont-Doherty Earth Observatory, New York, USA

3 Centre de Documentation et de Recherche sur l’Art et les Traditions Orales à Madagascar, Université de Toliara, Madagascar

4 The Morombe Archaeological Project, Morombe, Madagascar

5 Max Planck Institute for Geoanthropology, Jena, Germany

\* Author for correspondence: ✉ dsd2149@columbia.edu

† Deceased

**Detailed methods**

*Network analysis*

The incomplete nature of archaeological data holds significant challenges to the implementation of social network analysis (SNA) methods for understanding the past (Brughmans 2013; Mills 2017; Peeples 2017; Roberts *et al.* 2021). Various solutions have been proposed around the common theme of statistical validation to assess possible biases or errors in the data being analysed (Peeples & Roberts 2013; Östborn & Gerding 2014). Östborn & Gerting (2014) discuss the need for statistical rigour in network analyses in archaeology and advocate for a random permutation approach to randomly reshuffle data to evaluate observed patterns from randomly dispersed datasets.

Brughmans (2013) argues that there are two major problems with many recent implementations of SNA in archaeology: 1) a general unawareness of the history and diversity of formal network methods and their archaeological suitability has resulted in a very limited scope of SNA applications; and 2) most applications of SNA in archaeology are not driven by research questions, but rather a limited number of popular models and techniques. Brughmans (2013) suggests that framing studies of archaeological SNA applications using complex systems theory can help alleviate some of the limitations.

Central tenets of SNA are that: 1) Actors and their actions are viewed as interdependent; 2) ties or linkages between actors are channels for the transfer of resources; 3) network models view the network structural environment as providing opportunities for, or constraints on, individual action; and 4) network models conceptualise structure (social, economic, political, and so forth) as lasting patterns of relations among actors (Wasserman & Faust 1994; Brughmans 2013).

*Network indices*

We tested three commonly used comparative indices: co-presence, Brainerd-Robinson (BR) similarity (Brainerd 1951; Robinson 1951), and chi-square distance. Co-presence is a simple similarity metric that establishes connections based on the presence of particular categories of data at multiple sites (Brughmans 2010). Following Peeples (2017), co-presence is calculated as: 𝑃=𝐴×𝐴𝑇

Where P is the number of overlapping categories between sites, A is the incidence matrix of categories, and AT is the transposed matrix of those categories. We generated co-presence networks using a threshold of 50 per cent similarity. This threshold was chosen based on trial-and-error, whereby 50 per cent yielded the best results.

BR similarity calculates similarity between nodes as a proportion of the representation of the total number of categories present within the data. This is a commonly applied similarity metric, and is calculated, following Peeples (2017), using the equation: 𝑆=2−Σ𝑐|𝑥𝑐−𝑦𝑐|2

Where *S* is the BR similarity score, *c* represents all the categories of data, *x* is the proportion of *c* in the first data assemblage, *y* is the proportion of *c* in the second assemblage.

Lastly, chi-square distance is a measurement used for correspondence analyses that is weighted by the inverse of a data category’s frequency (Dodge 2008). Chi-square distance is calculated using the equation: 𝑋𝑛𝑐=√Σ1𝑎𝑛(𝑥𝑛−𝑦𝑛)2

Where *an* is the proportional abundance of the *n*th element of the average row profile in the data, and *x* and *y* represent the row profiles for the two sites being compared. Chi-square distances are useful for accounting for rare attributes in the formation of data connections (Peeples & Roberts 2013).

*Assessment of sampling error on network results*

To assess the effects that sampling error may have on our results, we calculated centrality metrics (degree, eigenvector, and betweenness) using 1000 bootstrap simulations to re-sample our data (following Mills *et al.* 2013; also see Roberts *et al.* 2021) and evaluate changes between randomized samples and our original dataset. Increased variability indicates higher risk of sampling error. Degree centrality for a node is defined as the total number of direct connections in which that node is involved (Peeples & Roberts 2013; Peeples 2017). Betweenness centrality is defined as the number of shortest paths between pairs of nodes in a network involving the target node divided by the total number of shortest paths in the network as a whole (Peeples & Roberts 2013; Peeples 2017). Eigenvector centrality is a measure of a node’s importance in a network defined in relation to other nodes to which it is connected (Peeples & Roberts 2013; Peeples 2017; Roberts *et al*. 2021).

Next, we re-assessed these networks for their resilience to sampling biases using 1000 bootstrap simulations to subsample the data into 10 per cent intervals and calculate the rank-order correlation (Spearman’s ρ) of the overall sample and each sub-sample (Costenbader & Valente 2003; Peeples 2017). We also assessed these biases using fewer numbers of simulations (100, 200, 500), and results remained largely identical. This allows us to evaluate the errors in the dataset that may arise from sampling issues (see supplemental code at https://github.com/d-davis/mada\_ceramic). This procedure is performed to account for missing nodes and edges in the dataset, which often plague archaeological investigations.

Then, we assessed the stability of individual nodes and edges in the network by using 1000 bootstrapped simulations of our network data to create sub-sampled datasets. This allowed us to compare the original dataset with sub-sampled components for agreement or divergence.

*Ceramic chronologies*

Relative chronologies for ceramics follow the typologies described in Douglass (2016). Based upon prior observations and studies (e.g. Wright *et al.* 1996, Parker Pearson 2010, Douglass 2016; Hixon *et al.* 2021), ceramics containing triangular punctation marks and incising were found among the oldest archaeological contexts, spanning from the ninth century AD to between the thirteenth and sixteenth centuries AD. Circular and square punctations appear slightly later (around the eleventh century), and the latest decorative style is shell-combing, which becomes prevalent around the eighteenth to twentieth centuries. Using these decorative characteristics, we constructed the relative chronology used in this analysis.

*Palaeoclimate assessment using Bayesian Change Point Analysis*

We used Bayesian change point analysis (BCPA; Erdman & Emerson 2007), following Hixon *et al*. (2021) to estimate general trends in climatological conditions from speleothem proxies collected by Faina *et al.* (2021) in Asafora Cave, south-west Madagascar. BCPA is a statistical modelling approach that uses Markov Chain simulation to identify splits in a sequence of datapoints that can be approximated reasonably with a single mean value. We conduct BCPA in *R* (v. 4.1.2; R Core Team 2021) using the *bcp* package (Erdman & Emerson 2008).

**References**

Brainerd, G.W. 1951. The place of chronological ordering in archaeological analysis. *American Antiquity* 16: 301–13. <https://doi.org/10.2307/276979>

Brughmans, T. 2010. Connecting the dots: towards archaeological network analysis*. Oxford Journal of Archaeology* 29: 277–303. <https://doi.org/10.1111/j.1468-0092.2010.00349.x>

Brughmans, T. 2013. Thinking through networks: a review of formal network methods in archaeology. *Journal of Archaeological Method and Theory* 20: 623–62. https://doi.org/10.1007/s10816-012-9133-8

Costenbader, E. & T.W. Valente. 2003. The stability of centrality measures when networks are sampled. *Social Networks* 25: 283–307. <https://doi.org/10.1016/S0378-8733(03)00012-1>

Dodge, Y. 2008. Chi-square distance, in Y. Dodge (ed.) *The concise encyclopedia of statistics*: 68–70. New York: Springer. https://doi.org/10.1007/978-0-387-32833-1\_53

Douglass, K. 2016. An archaeological investigation of settlement and resource exploitation patterns in the Velondriake Marine Protected Area, Southwest Madagascar, ca. 900 BC to AD 1900. Unpublished PhD Dissertation, Yale University.

Erdman, C. & J.W. Emerson. 2008. bcp: an R package for performing a Bayesian analysis of change point problems. *Journal of Statistical Software* 23: 1–13. <https://doi.org/10.18637/jss.v023.i03>

Faina, P., S.J. Burns, L.R. Godfrey, B.E. Crowley, N. Scroxton, D. McGee, M.R. Sutherland & L. Ranivoharimanana. 2021. Comparing the paleoclimates of northwestern and southwestern Madagascar during the late Holocene: implications for the role of climate in megafaunal extinction. *Malagasy Nature* 15: 108–127.

Hixon, S.W., J.H. Curtis, M. Brenner, K.G. Douglass, A.I. Domic, B.J. Culleton, S.J. Ivory & D.J. Kennett. 2021a. Drought coincided with, but does not explain, late holocene megafauna extinctions in SW Madagascar. *Climate* 9: 138. <https://doi.org/10.3390/cli9090138>

Mills, B.J. 2017. Social network analysis in archaeology. *Annual Review of Anthropology* 46: 379–97. https://doi.org/10.1146/annurev-anthro-102116-041423

Östborn, P. & H. Gerding. 2014. Network analysis of archaeological data: a systematic approach. *Journal of Archaeological Science* 46: 75–88. https://doi.org/10.1016/j.jas.2014.03.015

Parker Pearson, M. 2010. *Pastoralists, warriors and colonists: the archaeology of southern Madagascar.* Oxford: Archaeopress. <https://doi.org/10.30861/9781407306803>

Peeples, M.A. 2017. Network science and statistical techniques for dealing with uncertainties in archaeological datasets. Available at http://www.mattpeeples.net/netstats.html (accessed 12.30.20).

Peeples, M.A. & J.M. Roberts. 2013. To binarize or not to binarize: relational data and the construction of archaeological networks. *Journal of Archaeological Science* 40: 3001–10. h[ttps://doi.org/10.1016/j.jas.2013.03.014](https://doi.org/10.1016/j.jas.2013.03.014)

R Core Team. 2020. *R:* *a language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. Available at <https://www.r-project.org/>

Roberts, J.M., Y. Yin, E. Dorshorst, M.A. Peeples, B.J. Mills. 2021. Assessing the performance of the bootstrap in simulated assemblage networks. *Social Networks* 65: 98–109. https://doi.org/10.1016/j.socnet.2020.11.005

Robinson, W.S. 1951. A method for chronologically ordering archaeological deposits. *American Antiquity* 16: 293–301. <https://doi.org/10.2307/276978>

Wasserman, S. & K. Faust. 1994. *Social network analysis. Structural analysis in the social sciences.* Cambridge: Cambridge University Press. <https://doi.org/10.1017/CBO9780511815478>

Wright, H.T., P. Vérin, Ramilisonina, D. Burney, L.P. Burney & K. Matsumoto. 1996. The evolution of settlement systems in the Bay of Boeny and the Mahavavy River Valley, north-western Madagascar. *AZANIA* 31: 37–73. <https://doi.org/10.1080/00672709609511456>

**Code for analyses undertaken**

This R Markdown document includes the code necessary to replicate the analysis in the associated manuscript. In what follows, code and its associated description will be provided along with the output of each component of the analysis.

The following code implements a network analysis of ceramic data from south-west Madagascar following the protocol developed by Peeples (2017). The original script can be found at: <http://www.mattpeeples.net/netstats.html>

##Load libraries and necessary datasets

|  |
| --- |
| *#Load required libraries* library(statnet) *#for network analysis* library(tnet) *#for network analysis* library(rgdal) *#for mapping GIS files* library(here)  *#set working directory* setwd(here())  *#Load data as properly formatted CSV files (See Peeples 2017 for formatting information)*  *# the name of each row (site name) should be the first column in the input ta ble*  d\_data1 <- read.table(file='Early\_Period\_Data.csv', sep=',', header=T, row.na mes=1)  d\_data2 <- read.table(file='Middle\_Period\_Data.csv', sep=',', header=T, row.n ames=1)  d\_data3 <- read.table(file='Late\_Period\_Data.csv', sep=',', header=T, row.nam es=1)    *#Load shapefile of study area*  AOI <- readOGR(dsn='Velondriake\_AOI.shp')  ## OGR data source with driver: ESRI Shapefile  ## Source: "[path to file]" |

## with 1 features

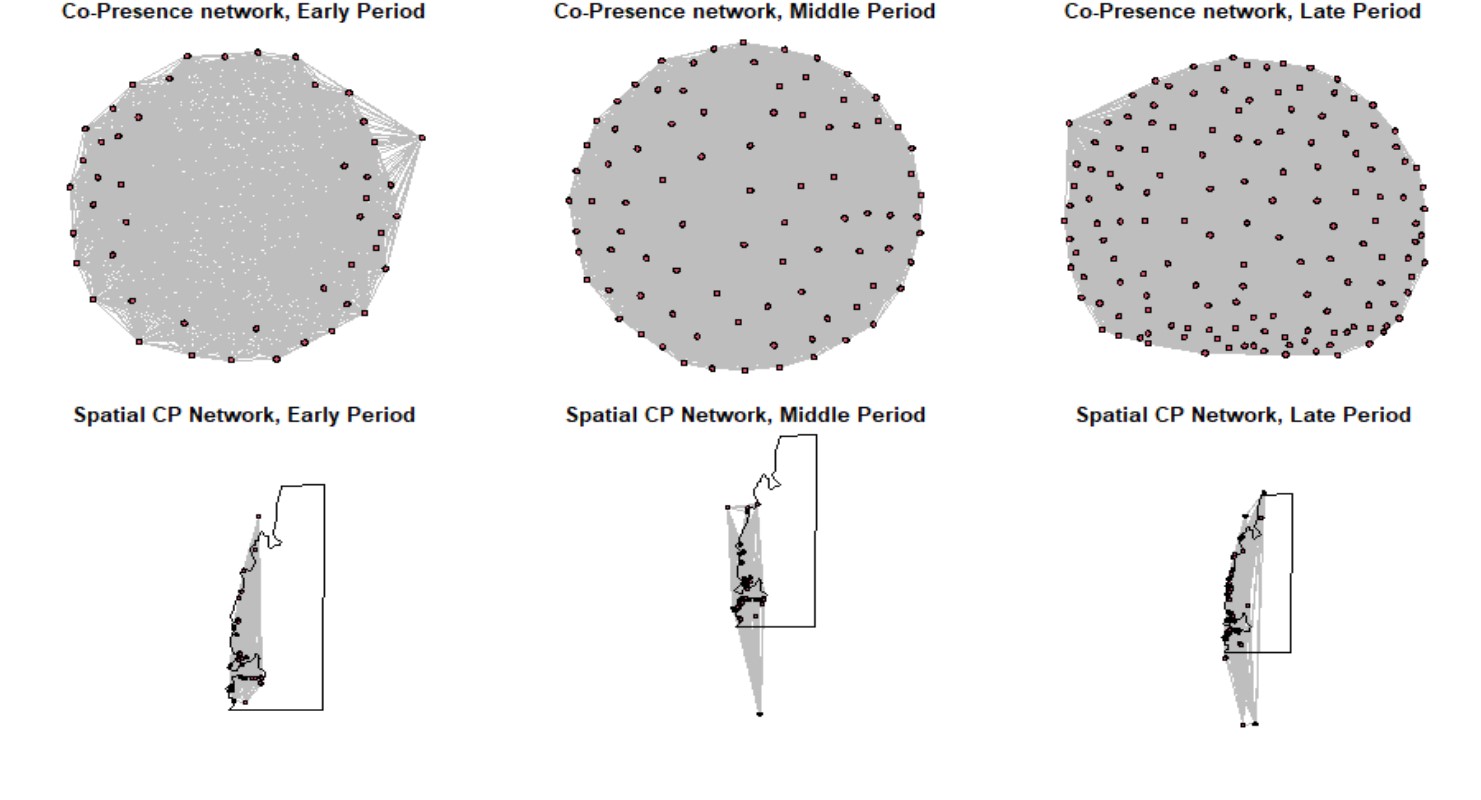
## It has 1 fields

##Implement functions for co-presence, Brainerd-Robinson (BR) coefficient, and chisquare distance metrics.

|  |
| --- |
| *#Function to create a co-occurance dataset using presence/absence*  co.p <- **function**(x, thresh = 0.1) {  *# create matrix of proportions from ceramic data*  temp <- prop.table(as.matrix(x), 1)  *# define anything with greater than or equal to 0.1 as present (1)*  temp[temp >= thresh] <- 1  *# define all other cells as absent (0)*  temp[temp < 1] <- 0  *# matrix algebraic calculation to find co-occurence (%\*% indicates matrix*  *# multiplication)*  out <- temp %\*% t(temp) return(out)  }    *# run the function on the datasets* d\_data1P <- co.p(d\_data1) *#Decoration data* d\_data2P <- co.p(d\_data2) *#Manufacturing data* d\_data3P <- co.p(d\_data3)    *# Function for calculating Brainerd-Robinson (BR) coefficients # This creates a matrix of similarity values used in many SNAs* sim.mat <- **function**(x) { *# get names of sites*  names <- row.names(x)  x <- na.omit(x) *# remove any rows with missing data*  x <- prop.table(as.matrix(x), 1) *# convert to row proportions*  rd <- dim(x)[1]  *# create an empty symmetric matrix of 0s*  results <- matrix(0, rd, rd)  *# the following dreaded double for-loop goes through every cell in the*  *# output data table and calculates the BR value as descried above*  **for** (s1 **in** 1:rd) { **for** (s2 **in** 1:rd) {  x1Temp <- as.numeric(x[s1, ]) x2Temp <- as.numeric(x[s2, ])  results[s1, s2] <- 2 - (sum(abs(x1Temp - x2Temp)))  } }  row.names(results) <- names *# assign row names to output*  colnames(results) <- names *# assign column names to output*  results <- results/2 *# rescale results between 0 and 1*  results <- round(results, 3) *# round results*  return(results)  } *# return the final output table* |

|  |
| --- |
| *# Run the BR coefficient function on our sample data* d\_data1BR <- sim.mat(d\_data1) *#Decoration data* d\_data2BR <- sim.mat(d\_data2) *#Manufacturing data* d\_data3BR <- sim.mat(d\_data3)    *# Chi-square (X2) distance function* chi.dist <- **function**(x) {  rowprof <- x/apply(x, 1, sum) *# calculates the profile for every row*  avgprof <- apply(x, 2, sum)/sum(x) *# calculates the average profile*  *# creates a distance object of $\chi^{2}$ distances*  chid <- dist(as.matrix(rowprof) %\*% diag(1/sqrt(avgprof)))  *# return the reults*  return(as.matrix(chid))  }  *# Run the X2 function and then create the rescaled 0-1 version* d\_data1X <- chi.dist(d\_data1) *#Decoration data* ## Warning in sqrt(avgprof): NaNs produced  d\_data1X01 <- d\_data1X/max(d\_data1X)  d\_data2X <- chi.dist(d\_data2)*#Manufacturing data* ## Warning in sqrt(avgprof): NaNs produced  d\_data2X01 <- d\_data2X/max(d\_data2X)  d\_data3X <- chi.dist(d\_data3) *#Decoration data* ## Warning in sqrt(avgprof): NaNs produced  d\_data3X01 <- d\_data3X/max(d\_data3X)      ***###VISIUALIZING NETWORKS***    *# create network object from co-occurrence*  Pnet\_d1 <- network(d\_data1P, directed = F)  Pnet\_d2 <- network(d\_data2P, directed = F)  Pnet\_d3 <- network(d\_data3P, directed = F)  *# Now let's add names for our nodes based on the row names of our original # matrix*  Pnet\_d1 %v% "vertex.names" <- row.names(d\_data1P)  Pnet\_d2 %v% "vertex.names" <- row.names(d\_data2P)  Pnet\_d3 %v% "vertex.names" <- row.names(d\_data3P)  *# look at the results*  Pnet\_d1 |

|  |  |
| --- | --- |
| ## Network attributes:  ## vertices = 48  ## directed = FALSE  ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 1119  ## missing edges= 0  ## non-missing edges= 1119  ##  ## Vertex attribute names:  ## vertex.names  ##  ## Edge attribute names not shown  Pnet\_d2  ## Network attributes:  ## vertices = 86  ## directed = FALSE  ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 3655  ## missing edges= 0  ## non-missing edges= 3655  ##  ## Vertex attribute names:  ## vertex.names  ##  ## Edge attribute names not shown  Pnet\_d3  ## Network attributes:  ## vertices = 146  ## directed = FALSE  ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 10559  ## missing edges= 0  ## non-missing edges= 10559  ##  ## Vertex attribute names:  ## vertex.names  ##  ## Edge attribute names not shown | |
| *# plot network using default layout* par(mfrow = c(2, 3))par(mar=c(0.5,1,1,0.5))    plot(Pnet\_d1, edge.col = "gray", edge.lwd = 0.10, vertex.cex = 0.75, main = " Co-Presence network, Early Period")  plot(Pnet\_d2, edge.col = "gray", edge.lwd = 0.10, vertex.cex = 0.75, main = " Co-Presence network, Middle Period")  plot(Pnet\_d3, edge.col = "gray", edge.lwd = 0.10, vertex.cex = 0.75, main = "  Co-Presence network, Late Period")  *# plot network spatially using geographic coordinates*  plot(Pnet\_d1, edge.col = "gray", edge.lwd = 0.10, vertex.cex = 0.5, main = "S patial CP Network, Early Period",coord = d\_data1[,10:11])plot(AOI, add=T) *#overlay network with coastline of study area*  plot(Pnet\_d2, edge.col = "gray", edge.lwd = 0.10, vertex.cex = 0.5, main = "S patial CP Network, Middle Period",coord = d\_data2[,10:11])plot(AOI, add=T) *#overlay network with coastline of study area*    plot(Pnet\_d3, edge.col = "gray", edge.lwd = 0.10, vertex.cex = 0.5, main = "S patial CP Network, Late Period",coord = d\_data3[,10:11]) | |
| plot(AOI, add=T) *#overlay network with coastline of study area* |  |



par(mfrow = c(1, 1)) *# return to single plotting mode*

*# Define our binary network object from BR similarity*

BRnet\_d1 <- network(event2dichot(d\_data1BR, method = "absolute", thresh = 0.6

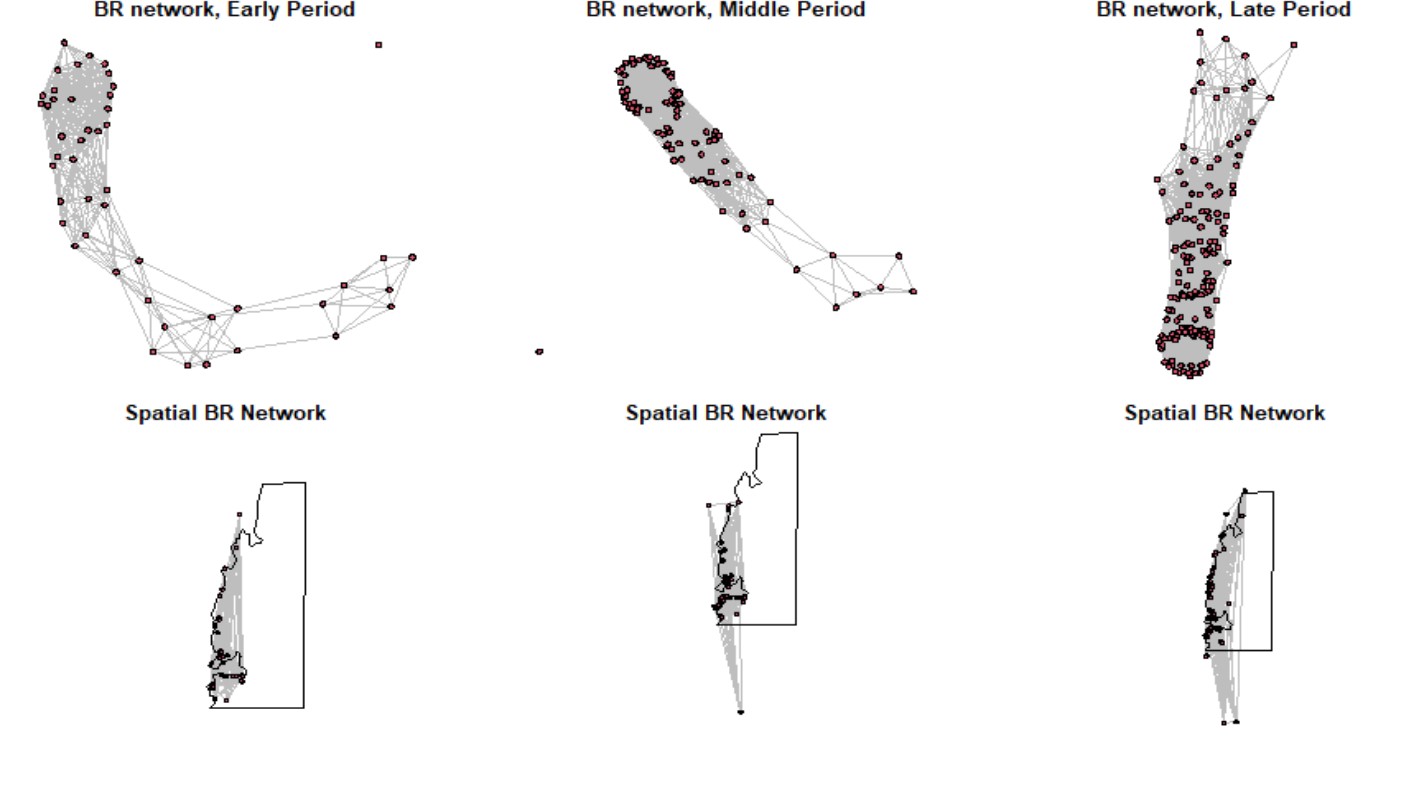
),

directed = F)

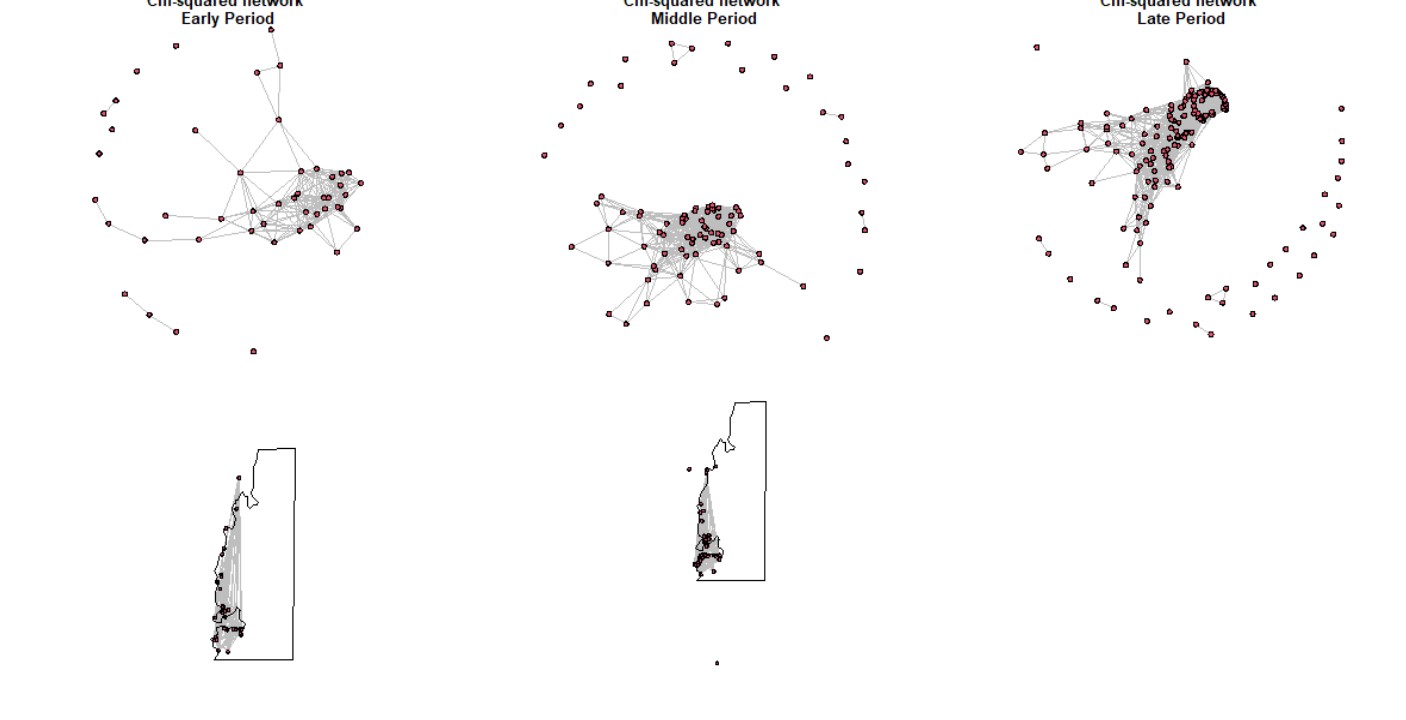
BRnet\_d2 <- network(event2dichot(d\_data2BR, method = "absolute", thresh = 0.6

|  |
| --- |
| ),  directed = F)  BRnet\_d3 <- network(event2dichot(d\_data3BR, method = "absolute", thresh = 0.6  ),  directed = F)  *# Add names for nodes based on the row names of the original matrix*  BRnet\_d1 %v% "vertex.names" <- row.names(d\_data1BR)  BRnet\_d2 %v% "vertex.names" <- row.names(d\_data2BR)  BRnet\_d3 %v% "vertex.names" <- row.names(d\_data3BR)  *# look at the results.*  BRnet\_d1  ## Network attributes:  ## vertices = 48  ## directed = FALSE  ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 351  ## missing edges= 0  ## non-missing edges= 351  ##  ## Vertex attribute names:  ## vertex.names  ##  ## No edge attributes  BRnet\_d2  ## Network attributes:  ## vertices = 86  ## directed = FALSE  ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 1743  ## missing edges= 0  ## non-missing edges= 1743  ##  ## Vertex attribute names:  ## vertex.names  ##  ## Edge attribute names not shown  BRnet\_d3  ## Network attributes:  ## vertices = 146  ## directed = FALSE |

|  |  |
| --- | --- |
| ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 4068  ## missing edges= 0  ## non-missing edges= 4068  ##  ## Vertex attribute names:  ## vertex.names  ##  ## Edge attribute names not shown  *# plot network using default layout* par(mfrow = c(2, 3))par(mar=c(0.5,1,1,0.5))    plot(BRnet\_d1, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.75, main =  "BR network, Early Period")  plot(BRnet\_d2, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.75, main =  "BR network, Middle Period")  plot(BRnet\_d3, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.75, main =  "BR network, Late Period")  *# plot network spatially using geographic coordinates*  plot(BRnet\_d1, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.5, main =  "Spatial BR Network",coord = d\_data1[,10:11])  plot(AOI, add=T) *#overlay network with coastline of study area*  plot(BRnet\_d2, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.5, main =  "Spatial BR Network",coord = d\_data2[,10:11])  plot(AOI, add=T) *#overlay network with coastline of study area*  plot(BRnet\_d3, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.5, main = "Spatial BR Network",coord = d\_data3[,10:11]) | |
| plot(AOI, add=T) *#overlay network with coastline of study area* |  |



|  |  |
| --- | --- |
| par(mfrow = c(1, 1)) *# return to single plotting mode*      *# Plot X2 distance similarity index*  *# This uses the 1 minus dataX01 calculation to convert*  *# X2 distance to a similarity (following Peeples 2017)*  Xnet\_d1 <- network(event2dichot(1 - d\_data1X01, method = "quantile", thresh =  0.8),  directed = F)  Xnet\_d2 <- network(event2dichot(1 - d\_data2X01, method = "quantile", thresh =  0.8),  directed = F)  Xnet\_d3 <- network(event2dichot(1 - d\_data3X01, method = "quantile", thresh =  0.8),  directed = F)    *# Once again add vertex names*  Xnet\_d1 %v% "vertex.names" <- row.names(d\_data1X01)  Xnet\_d2 %v% "vertex.names" <- row.names(d\_data2X01)  Xnet\_d3 %v% "vertex.names" <- row.names(d\_data3X01)  *# look at the results*  Xnet\_d1  ## Network attributes:## vertices = 48  ## directed = FALSE  ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 206 | |
| ## missing edges= 0  ## non-missing edges= 206  ##  ## Vertex attribute names:  ## vertex.names  ##  ## No edge attributes  Xnet\_d2  ## Network attributes:  ## vertices = 86  ## directed = FALSE  ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 696  ## missing edges= 0  ## non-missing edges= 696  ##  ## Vertex attribute names:  ## vertex.names  ##  ## No edge attributes  Xnet\_d3  ## Network attributes:  ## vertices = 146  ## directed = FALSE  ## hyper = FALSE  ## loops = FALSE  ## multiple = FALSE  ## bipartite = FALSE  ## total edges= 2058  ## missing edges= 0  ## non-missing edges= 2058  ##  ## Vertex attribute names:  ## vertex.names  ##  ## Edge attribute names not shown  *# plot network using default layout* par(mfrow = c(2, 3))par(mar=c(0.5,1,1,0.5))    plot(Xnet\_d1, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.75, main =  "Chi-squared network \nEarly Period")  plot(Xnet\_d2, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.75, main = | |
| "Chi-squared network \nMiddle Period")  plot(Xnet\_d3, edge.col = "gray", edge.lwd = 0.001, vertex.cex = 0.75, main = "Chi-squared network \nLate Period")    *# plot network using geographic coordinates*  plot(Xnet\_d1, edge.col = "gray", edge.lwd = 0.75, vertex.cex = 0.5, coord = d \_data1[,    10:11])  plot(AOI, add=T) *#overlay network with coastline of study area*  plot(Xnet\_d2, edge.col = "gray", edge.lwd = 0.75, vertex.cex = 0.5, coord = d \_data2[,    10:11])  plot(AOI, add=T) *#overlay network with coastline of study*    plot(Xnet\_d3, edge.col = "gray", edge.lwd = 0.75, vertex.cex = 0.5, coord = d \_data3[,  10:11]) | |
| plot(AOI, add=T) *#overlay network with coastline of study area* |  |



par(mfrow = c(1, 1))

#Peeples (2017) finds that weighted networks do not perform well with similarity or distance matrices, so we do not use them here.

## CALCULATE RAW DATAVALUES OF NETWORK CONNECTIVITY

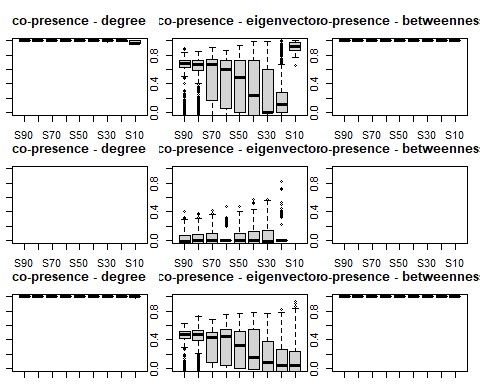
*# Calculate centrality scores for binary networks* net.stats <- **function**(y) {

*# calculate degree centrality*

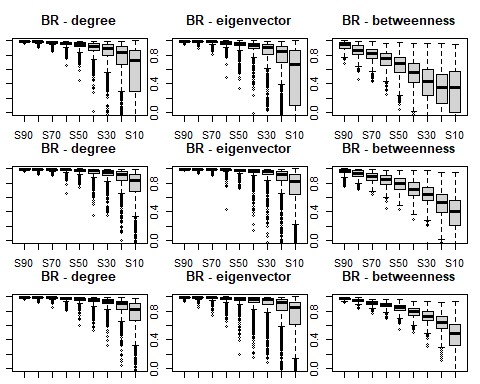
dg <- as.matrix(sna::degree(y, gmode = "graph"))

|  |
| --- |
| *# calculate and scale eigenvector centrality*  eg <- as.matrix(sna::evcent(y, use.eigen = TRUE)) eg <- sqrt((eg^2) \* length(eg)) *# calculate betweenness centrality*  bw <- sna::betweenness(y, gmode = "graph") *# combine centrality scores into matrix*  output <- cbind(dg, eg, bw)  rownames(output) <- rownames(as.matrix(y)) colnames(output) <- c("dg", "eg", "bw") return(output)  } *# return results of this function*    *# net stats for binary co-presence network* co.p.stats\_d <- net.stats(Pnet\_d1)co.p.stats\_d2 <- net.stats(Pnet\_d2)co.p.stats\_d3 <- net.stats(Pnet\_d3)  *# net stats for binary BR similarity network*  BR.stats\_d <- net.stats(BRnet\_d1)  BR.stats\_d2 <- net.stats(BRnet\_d2)  BR.stats\_d3 <- net.stats(BRnet\_d3)    *# net stats for binary X^2 similarity network (1-distance)*  X.stats\_d <- net.stats(Xnet\_d1)  X.stats\_d2 <- net.stats(Xnet\_d2)  X.stats\_d3 <- net.stats(Xnet\_d3)head(X.stats\_d)  ## dg eg bw  ## G-11-20 1 2.480331e-05 0  ## VATO Za003 0 3.076740e-15 0  ## G-48-20 2 3.797514e-04 36  ## LSS\_52 3 1.534534e-02 36  ## GI118 2 1.528389e-02 0  ## GI128 1 1.002275e-03 0  *#write.csv(X.stats\_d3, "X2\_1700\_1900\_Stats.csv")*  *##RUN GRAPH LEVEL CENTRALIZATION METRICS*  detach(package:tnet, unload=TRUE) # unload tnet package (required for this an alysis)  *#co-presence network*  centralization(Pnet\_d1,degree,normalize=T) #calculate binary degree centraliz ation  ## centralization(Pnet\_d1,degree,normalize=T) #calculate binary degree centra lization  ## [1] 0.008325624 |
| centralization(Pnet\_d2,betweenness,normalize=T) #calculate binary betweenness centralization  ## centralization(Pnet\_d2,betweenness,normalize=T) #calculate binary betweenn ess centralization  ## [1] 0    centralization(Pnet\_d3,evcent,normalize=T) #calculate binary eigenvector cent ralization  ## centralization(Pnet\_d3,evcent,normalize=T) #calculate binary eigenvector c entralization ## [1] 0.0001854757    *#BR network*  centralization(BRnet\_d1,degree,normalize=T) #calculate binary degree centrali zation  ## centralization(BRnet\_d1,degree,normalize=T) #calculate binary degree centr alization ## [1] 0.2303423  centralization(BRnet\_d2,betweenness,normalize=T) #calculate binary betweennes s centralization  ## centralization(BRnet\_d2,betweenness,normalize=T) #calculate binary between ness centralization ## [1] 0.06405785  centralization(BRnet\_d3,evcent,normalize=T) #calculate binary eigenvector cen tralization  ## centralization(BRnet\_d3,evcent,normalize=T) #calculate binary eigenvector centralization ## [1] 0.0561337 *# Chi-Square Network*  centralization(Xnet\_d1,degree,normalize=T) #calculate binary degree centraliz ation  ## centralization(Xnet\_d1,degree,normalize=T) #calculate binary degree centra lization  ## [1] 0.2978723  centralization(Xnet\_d2,betweenness,normalize=T) #calculate binary betweenness centralization  ## centralization(Xnet\_d2,betweenness,normalize=T) #calculate binary betweenn ess centralization  ## [1] 0.03032848 |
| centralization(Xnet\_d3,evcent,normalize=T) #calculate binary eigenvector cent ralization  ## centralization(Xnet\_d3,evcent,normalize=T) #calculate binary eigenvector c entralization ## [1] 0.103616  *# The following function does the same calculation as above but is set up to*  *# work with the output of net.stats and net.stats.wt* nsim <- 1000    samp.frac <- c("S90", "S80", "S70", "S60", "S50", "S40", "S30", "S20", "S10")    cv.resamp.bin <- **function**(x) {  *# calculate all network stats for the original network*  stats.g <- net.stats(x) mat <- as.matrix(x)  dim.x <- dim(mat)[1] *# count number of rows (nodes)*  *# define empty matrices for output*  dg.mat <- matrix(NA, nsim, 9) ev.mat <- matrix(NA, nsim, 9) bw.mat <- matrix(NA, nsim, 9)  *# add column names based on sampling fraction*  colnames(dg.mat) <- samp.frac colnames(ev.mat) <- samp.frac colnames(bw.mat) <- samp.frac    *# this double loop goes through each sampling fraction and each random*  *# replicate to cacluate centrality statistics and runs a Spearman's rho*  *# correlation between the resulting centrality values and the original*  *# sample*  **for** (j **in** 1:9) { **for** (i **in** 1:nsim) {  sub.samp <- sample(seq(1, dim.x), size = round(dim.x \* ((10 - j)/10),  0), replace = F)  temp.stats <- net.stats(mat[sub.samp, sub.samp])  dg.mat[i, j] <- suppressWarnings(cor(temp.stats[, 1], stats.g[sub.samp, 1], metho d = "spearman"))  ev.mat[i, j] <- suppressWarnings(cor(temp.stats[, 2], stats.g[sub.samp, 2], metho d = "spearman"))  bw.mat[i, j] <- suppressWarnings(cor(temp.stats[, 3], stats.g[sub.samp, 3], metho d = "spearman"))  } }  out.list <- list() *# create list for output and populate it*  out.list[[1]] <- dg.mat out.list[[2]] <- ev.mat |

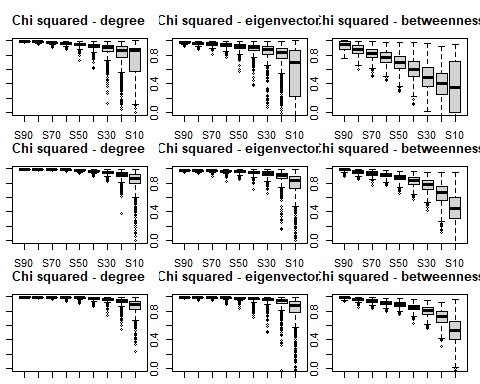
|  |  |
| --- | --- |
| out.list[[3]] <- bw.mat return(out.list)  } *# return the resulting list*  cop.rs\_d <- cv.resamp.bin(Pnet\_d1)cop.rs\_d2 <- cv.resamp.bin(Pnet\_d2)cop.rs\_d3 <- cv.resamp.bin(Pnet\_d3)BR.rs\_d <- cv.resamp.bin(BRnet\_d1)  BR.rs\_d2 <- cv.resamp.bin(BRnet\_d2)  BR.rs\_d3 <- cv.resamp.bin(BRnet\_d3)  X.rs\_d <- cv.resamp.bin(Xnet\_d1)  X.rs\_d2 <- cv.resamp.bin(Xnet\_d2)  X.rs\_d3 <- cv.resamp.bin(Xnet\_d3)    ***##PLOT DECORATIVE NETWORKS***  par(mfrow = c(3, 3)) *# set up for 3 by 3 plotting* par(mar=c(1,1,3,1))  *# plot boxplots by sampling fraction for each measure and each network type* boxplot(cop.rs\_d[[1]], ylim = c(0, 1), main = "co-presence - degree", xlab =  "sampling fraction",  ylab = "Spearmans rho")  boxplot(cop.rs\_d[[2]], ylim = c(0, 1), main = "co-presence - eigenvector", xl ab = "sampling fraction")  boxplot(cop.rs\_d[[3]], ylim = c(0, 1), main = "co-presence - betweenness", xl ab = "sampling fraction")  boxplot(cop.rs\_d2[[1]], ylim = c(0, 1), main = "co-presence - degree", xlab =  "sampling fraction",  ylab = "Spearmans rho")  boxplot(cop.rs\_d2[[2]], ylim = c(0, 1), main = "co-presence - eigenvector", x lab = "sampling fraction")  boxplot(cop.rs\_d2[[3]], ylim = c(0, 1), main = "co-presence - betweenness", x lab = "sampling fraction")  boxplot(cop.rs\_d3[[1]], ylim = c(0, 1), main = "co-presence - degree", xlab =  "sampling fraction",  ylab = "Spearmans rho")  boxplot(cop.rs\_d3[[2]], ylim = c(0, 1), main = "co-presence - eigenvector", x lab = "sampling fraction")  boxplot(cop.rs\_d3[[3]], ylim = c(0, 1), main = "co-presence - betweenness", x | |
| lab = "sampling fraction") |  |



|  |  |
| --- | --- |
| par(mfrow = c(3, 3)) *# set up for 3 by 3 plotting* par(mar=c(1,1,3,1))    boxplot(BR.rs\_d[[1]], ylim = c(0, 1), main = "BR - degree", xlab = "sampling fraction",  ylab = "Spearmans rho")  boxplot(BR.rs\_d[[2]], ylim = c(0, 1), main = "BR - eigenvector", xlab = "samp ling fraction")  boxplot(BR.rs\_d[[3]], ylim = c(0, 1), main = "BR - betweenness", xlab = "samp ling fraction")  boxplot(BR.rs\_d2[[1]], ylim = c(0, 1), main = "BR - degree", xlab = "sampling fraction",  ylab = "Spearmans rho")  boxplot(BR.rs\_d2[[2]], ylim = c(0, 1), main = "BR - eigenvector", xlab = "sam pling fraction")  boxplot(BR.rs\_d2[[3]], ylim = c(0, 1), main = "BR - betweenness", xlab = "sam pling fraction")  boxplot(BR.rs\_d3[[1]], ylim = c(0, 1), main = "BR - degree", xlab = "sampling fraction",  ylab = "Spearmans rho")  boxplot(BR.rs\_d3[[2]], ylim = c(0, 1), main = "BR - eigenvector", xlab = "sam pling fraction")  boxplot(BR.rs\_d3[[3]], ylim = c(0, 1), main = "BR - betweenness", xlab = "sam | |
| pling fraction") |  |

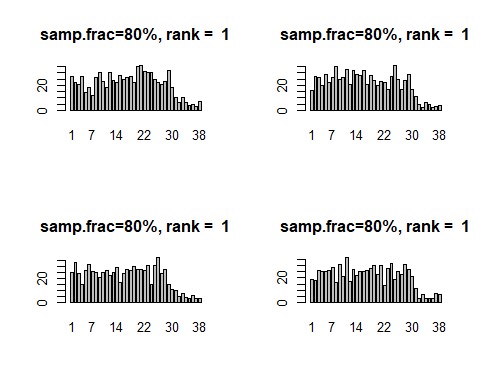


|  |  |
| --- | --- |
| par(mfrow = c(3, 3)) *# set up for 3 by 3 plotting* par(mar=c(1,1,3,1))    boxplot(X.rs\_d[[1]], ylim = c(0, 1), main = "Chi squared - degree", xlab = "s ampling fraction",  ylab = "Spearmans rho")  boxplot(X.rs\_d[[2]], ylim = c(0, 1), main = "Chi squared - eigenvector", xlab  = "sampling fraction")  boxplot(X.rs\_d[[3]], ylim = c(0, 1), main = "Chi squared - betweenness", xlab  = "sampling fraction")  boxplot(X.rs\_d2[[1]], ylim = c(0, 1), main = "Chi squared - degree", xlab = "  sampling fraction",  ylab = "Spearmans rho")  boxplot(X.rs\_d2[[2]], ylim = c(0, 1), main = "Chi squared - eigenvector", xla b = "sampling fraction")  boxplot(X.rs\_d2[[3]], ylim = c(0, 1), main = "Chi squared - betweenness", xla b = "sampling fraction")  boxplot(X.rs\_d3[[1]], ylim = c(0, 1), main = "Chi squared - degree", xlab = "  sampling fraction",  ylab = "Spearmans rho")  boxplot(X.rs\_d3[[2]], ylim = c(0, 1), main = "Chi squared - eigenvector", xla b = "sampling fraction")  boxplot(X.rs\_d3[[3]], ylim = c(0, 1), main = "Chi squared - betweenness", xla | |
| b = "sampling fraction") |  |

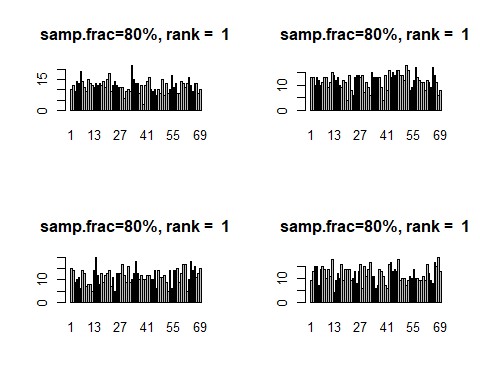


##Assess missing nodes within dataset

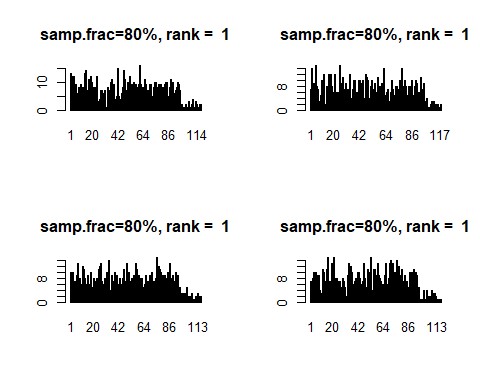
|  |
| --- |
| *#NODE ASSESSMENT*  nsim <- 1000 *#set number of replicates*    resamp.node <- **function**(x, samp.frac) { mat <- as.matrix(x) dim.x <- dim(mat)[1]  out.mat <- matrix(NA, dim.x, nsim) **for** (i **in** 1:nsim) {  sub.samp <- sample(seq(1, dim.x), size = round(dim.x \* samp.frac, 0),  replace = F)  *# calculate centrality statistic for a given sub-sample and put in output*  *# matrix*  temp.stats <- sna::degree(mat[sub.samp, sub.samp], gmode = "graph") out.mat[sub.samp, i] <- temp.stats  }  return(out.mat)  }  *#ASSESSMENT OF DECORATIVE NETWORKS*    *# calculate the rank order of degree centrality in the CP network* top.dg <- rank(-sna::degree(Pnet\_d1), ties.method = "min")par(mfrow = c(2, 2))  P.resamp <- resamp.node(Pnet\_d1, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates* |
| **for** (i **in** 1:4) {  barplot(table(apply(-P.resamp, 2, rank, ties.method = "random", na.last = " keep")[order(top.dg)[i],  ]), main = paste("samp.frac=80%, rank = ", top.dg[order(top.dg)[i]]))} |



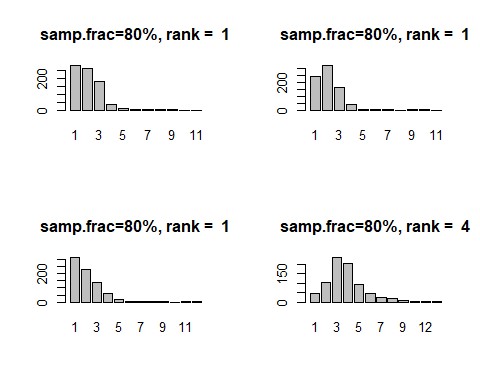
|  |
| --- |
| top.dg <- rank(-sna::degree(Pnet\_d2), ties.method = "min")par(mfrow = c(2, 2))  P.resamp <- resamp.node(Pnet\_d2, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots*  *# showing rank across all replicates* **for** (i **in** 1:4) {  barplot(table(apply(-P.resamp, 2, rank, ties.method = "random", na.last = " keep")[order(top.dg)[i],  ]), main = paste("samp.frac=80%, rank = ", top.dg[order(top.dg)[i]]))} |



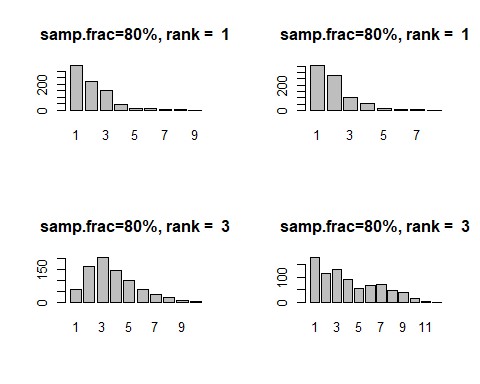
|  |
| --- |
| top.dg <- rank(-sna::degree(Pnet\_d3), ties.method = "min")par(mfrow = c(2, 2))  P.resamp <- resamp.node(Pnet\_d3, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates*    **for** (i **in** 1:4) {  barplot(table(apply(-P.resamp, 2, rank, ties.method = "random", na.last = " keep")[order(top.dg)[i],  ]), main = paste("samp.frac=80%, rank = ", top.dg[order(top.dg)[i]]))} |



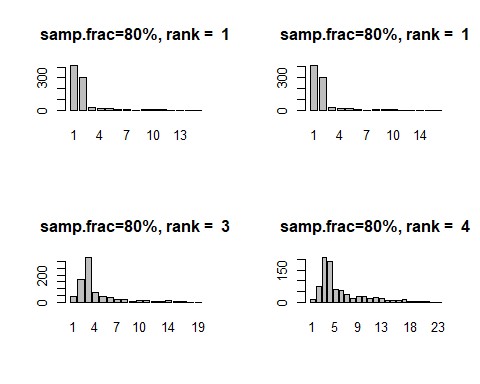
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(BRnet\_d1), ties.method = "min")  par(mfrow = c(2, 2))  BR.resamp <- resamp.node(BRnet\_d1, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates* |



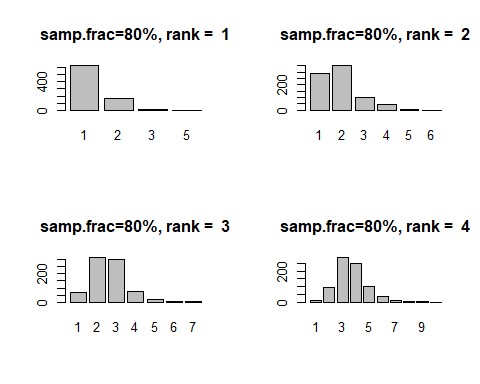
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(BRnet\_d2), ties.method = "min")  par(mfrow = c(2, 2))  BR.resamp <- resamp.node(BRnet\_d2, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates* |



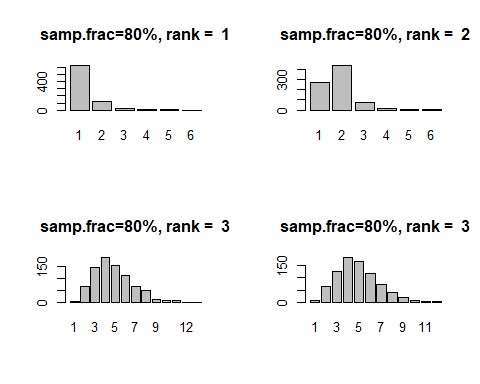
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(BRnet\_d3), ties.method = "min")  par(mfrow = c(2, 2))  BR.resamp <- resamp.node(BRnet\_d3, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates* |



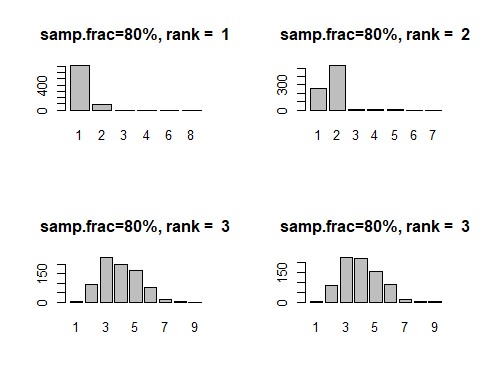
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(Xnet\_d1), ties.method = "min")par(mfrow = c(2, 2))  X.resamp <- resamp.node(Xnet\_d1, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates* |



|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(Xnet\_d2), ties.method = "min")par(mfrow = c(2, 2))  X.resamp <- resamp.node(Xnet\_d2, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates* |

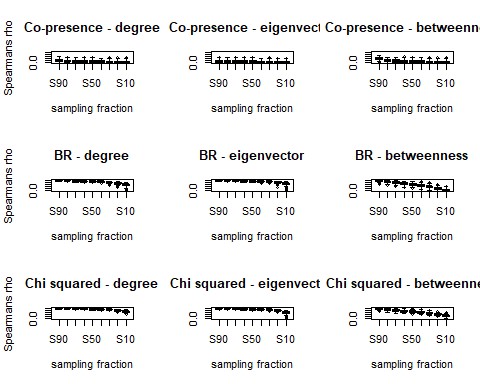


|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(Xnet\_d3), ties.method = "min")par(mfrow = c(2, 2))  X.resamp <- resamp.node(Xnet\_d3, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates* |

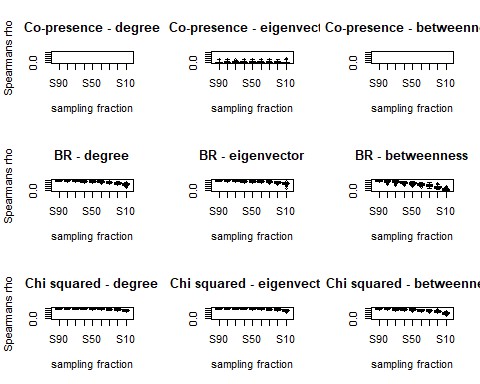


##Assess missing edges in dataset

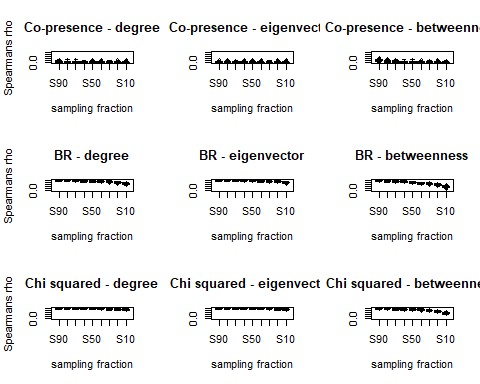
|  |
| --- |
| *# ASSESS MISSING EDGES*  *# set up to work with the output of net.stats* nsim <- 1000  samp.frac <- c("S90", "S80", "S70", "S60", "S50", "S40", "S30", "S20", "S10")  cv.resamp.edge <- **function**(x) { stats.g <- net.stats(x) mat <- as.matrix(x) dim.x <- dim(mat)[1] dg.mat <- matrix(NA, nsim, 9) ev.mat <- matrix(NA, nsim, 9) bw.mat <- matrix(NA, nsim, 9) colnames(dg.mat) <- samp.frac colnames(ev.mat) <- samp.frac colnames(bw.mat) <- samp.frac    **for** (j **in** 1:9) { **for** (i **in** 1:nsim) {  sub.samp <- sample(seq(1, network.edgecount(x)), size = round(network.e dgecount(x) \*  (j/10),  0), replace = F) temp.net <- x  net.reduced <- network::delete.edges(temp.net, sub.samp) temp.stats <- net.stats(net.reduced) |
| dg.mat[i, j] <- cor(temp.stats[, 1], stats.g[, 1], method = "spearman") ev.mat[i, j] <- cor(temp.stats[, 2], stats.g[, 2], method = "spearman") bw.mat[i, j] <- cor(temp.stats[, 3], stats.g[, 3], method = "spearman")  } }  out.list <- list() out.list[[1]] <- dg.mat out.list[[2]] <- ev.mat out.list[[3]] <- bw.mat return(out.list)  }    *#ASSESS DECORATIVE NETWORKS*    *# run the script for our three binary networks*  cop.edge.d <- cv.resamp.edge(Pnet\_d1)BR.edge.d <- cv.resamp.edge(BRnet\_d1)  X.edge.d <- cv.resamp.edge(Xnet\_d1)cop.edge.d2 <- cv.resamp.edge(Pnet\_d2)  BR.edge.d2 <- cv.resamp.edge(BRnet\_d2)  X.edge.d2 <- cv.resamp.edge(Xnet\_d2)cop.edge.d3 <- cv.resamp.edge(Pnet\_d3)BR.edge.d3 <- cv.resamp.edge(BRnet\_d3)  X.edge.d3 <- cv.resamp.edge(Xnet\_d3)  par(mfrow = c(3, 3))  boxplot(cop.edge.d[[1]], ylim = c(0, 1), main = "Co-presence - degree", xlab  = "sampling fraction",  ylab = "Spearmans rho")  boxplot(cop.edge.d[[2]], ylim = c(0, 1), main = "Co-presence - eigenvector", xlab = "sampling fraction")  boxplot(cop.edge.d[[3]], ylim = c(0, 1), main = "Co-presence - betweenness", xlab = "sampling fraction")  boxplot(BR.edge.d[[1]], ylim = c(0, 1), main = "BR - degree", xlab = "samplin g fraction",  ylab = "Spearmans rho")  boxplot(BR.edge.d[[2]], ylim = c(0, 1), main = "BR - eigenvector", xlab = "sa mpling fraction")  boxplot(BR.edge.d[[3]], ylim = c(0, 1), main = "BR - betweenness", xlab = "sa mpling fraction")  boxplot(X.edge.d[[1]], ylim = c(0, 1), main = "Chi squared - degree", xlab =  "sampling fraction",  ylab = "Spearmans rho")  boxplot(X.edge.d[[2]], ylim = c(0, 1), main = "Chi squared - eigenvector", xl ab = "sampling fraction")  boxplot(X.edge.d[[3]], ylim = c(0, 1), main = "Chi squared - betweenness", xl ab = "sampling fraction") |



|  |
| --- |
| par(mfrow = c(3, 3))  boxplot(cop.edge.d2[[1]], ylim = c(0, 1), main = "Co-presence - degree", xlab  = "sampling fraction",  ylab = "Spearmans rho")  boxplot(cop.edge.d2[[2]], ylim = c(0, 1), main = "Co-presence - eigenvector", xlab = "sampling fraction")  boxplot(cop.edge.d2[[3]], ylim = c(0, 1), main = "Co-presence - betweenness", xlab = "sampling fraction")  boxplot(BR.edge.d2[[1]], ylim = c(0, 1), main = "BR - degree", xlab = "sampli ng fraction",  ylab = "Spearmans rho")  boxplot(BR.edge.d2[[2]], ylim = c(0, 1), main = "BR - eigenvector", xlab = "s ampling fraction")  boxplot(BR.edge.d2[[3]], ylim = c(0, 1), main = "BR - betweenness", xlab = "s ampling fraction")  boxplot(X.edge.d2[[1]], ylim = c(0, 1), main = "Chi squared - degree", xlab =  "sampling fraction",  ylab = "Spearmans rho")  boxplot(X.edge.d2[[2]], ylim = c(0, 1), main = "Chi squared - eigenvector", x lab = "sampling fraction")  boxplot(X.edge.d2[[3]], ylim = c(0, 1), main = "Chi squared - betweenness", x lab = "sampling fraction") |

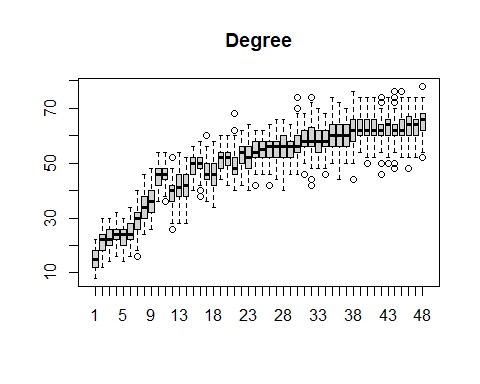


|  |
| --- |
| par(mfrow = c(3, 3))  boxplot(cop.edge.d3[[1]], ylim = c(0, 1), main = "Co-presence - degree", xlab  = "sampling fraction",  ylab = "Spearmans rho")  boxplot(cop.edge.d3[[2]], ylim = c(0, 1), main = "Co-presence - eigenvector", xlab = "sampling fraction")  boxplot(cop.edge.d3[[3]], ylim = c(0, 1), main = "Co-presence - betweenness", xlab = "sampling fraction")  boxplot(BR.edge.d3[[1]], ylim = c(0, 1), main = "BR - degree", xlab = "sampli ng fraction",  ylab = "Spearmans rho")  boxplot(BR.edge.d3[[2]], ylim = c(0, 1), main = "BR - eigenvector", xlab = "s ampling fraction")  boxplot(BR.edge.d3[[3]], ylim = c(0, 1), main = "BR - betweenness", xlab = "s ampling fraction")  boxplot(X.edge.d3[[1]], ylim = c(0, 1), main = "Chi squared - degree", xlab =  "sampling fraction",  ylab = "Spearmans rho")  boxplot(X.edge.d3[[2]], ylim = c(0, 1), main = "Chi squared - eigenvector", x lab = "sampling fraction")  boxplot(X.edge.d3[[3]], ylim = c(0, 1), main = "Chi squared - betweenness", x lab = "sampling fraction") |

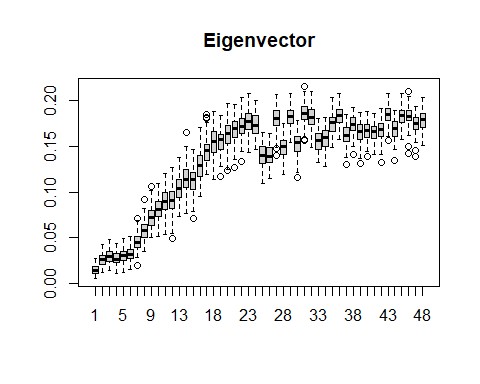


##Check for additional potential errors with missing data. This helps to validate our conclusions despite limited datasets

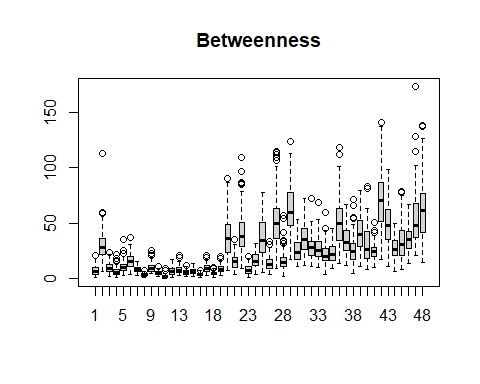
|  |
| --- |
| *###CHECK FOR SAMPLING ERRORS AND BIASES IN DATASET*  *## This code was originally written by Peeples (2017) and uses**## simulations to check for sampling issues*  nsim <- 100 *# we use 100 simulations due to computational limitations. We did assess the data using larger numbers of simulations (200, 500, 1000) and it provided identical results, but also occasionally crashed the computer used for analysis.*    net.prob <- **function**(x) { x <- as.matrix(x)  x[x < 0] <- 0 *# define threshold for excluding edges*  net.list <- list() **for** (i **in** 1:nsim) { y <- x  **for** (j **in** 1:length(x)) {  y[j] <- rbinom(1, 1, prob = x[j])  }  net.list[[i]] <- network(y, directed = F)  }  return(net.list)  }    *#ASSESS DECORATIVE NETWORKS* |
| *# Run the script on the BR similarity matrix*  BRprob <- net.prob(d\_data1BR)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data1BR), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(BRprob[[i]])  }  *# show boxplot of degree centrality sorted by the degree cent score in the*  *# original similarity matrix*  boxplot(t(dg.mat[order(rowSums(d\_data1BR)), ]), main = "Degree") |



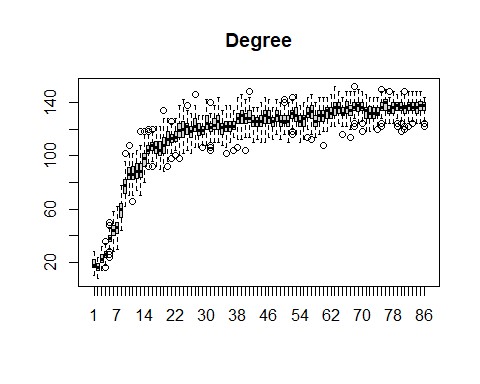
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data1BR), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(BRprob[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original similarity matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data1BR)), ]), main = "Eigenvector") |



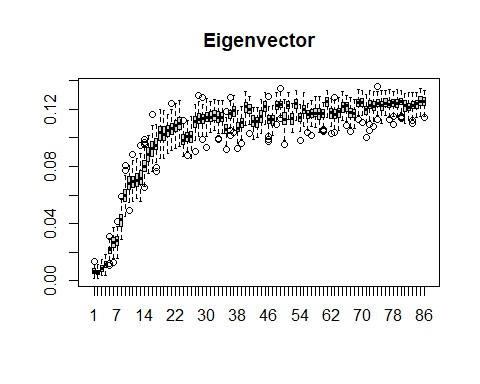
|  |  |
| --- | --- |
| bw.mat <- matrix(NA, nrow(d\_data1BR), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(BRprob[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original similarity matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data1BR)[, 2]), ]), main = "Betweennes s")  ## Warning in as.tnet(net, type = "weighted one-mode tnet"): There were selfloops | |
| ## in the edgelist, these were removed |  |



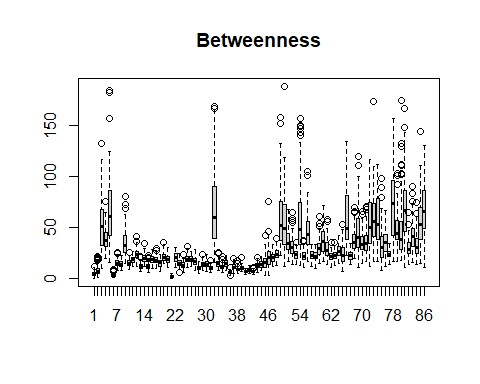
|  |
| --- |
| *# Run the script on the BR similarity matrix*  BRprob2 <- net.prob(d\_data2BR)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data2BR), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(BRprob2[[i]])  }  *# show boxplot of degree centrality sorted by the degree cent score in the*  *# original similarity matrix*  boxplot(t(dg.mat[order(rowSums(d\_data2BR)), ]), main = "Degree") |



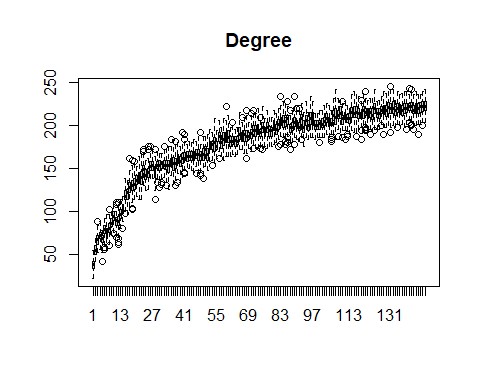
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data2BR), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(BRprob2[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original similarity matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data2BR)), ]), main = "Eigenvector") |



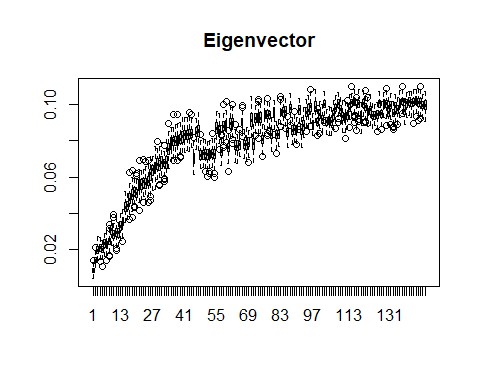
|  |  |
| --- | --- |
| bw.mat <- matrix(NA, nrow(d\_data2BR), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(BRprob2[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original similarity matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data2BR)[, 2]), ]), main = "Betweennes s")  ## Warning in as.tnet(net, type = "weighted one-mode tnet"): There were selfloops | |
| ## in the edgelist, these were removed |  |



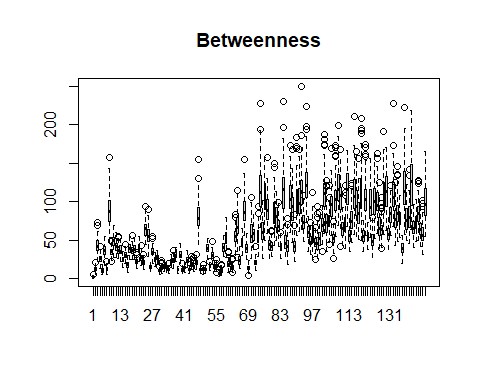
|  |
| --- |
| *# Run the script on the BR similarity matrix*  BRprob3 <- net.prob(d\_data3BR)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data3BR), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(BRprob3[[i]])  }  *# show boxplot of degree centrality sorted by the degree cent score in the*  *# original similarity matrix*  boxplot(t(dg.mat[order(rowSums(d\_data3BR)), ]), main = "Degree") |



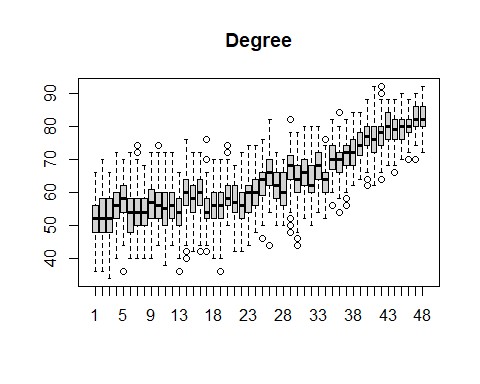
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data3BR), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(BRprob3[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original similarity matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data3BR)), ]), main = "Eigenvector") |



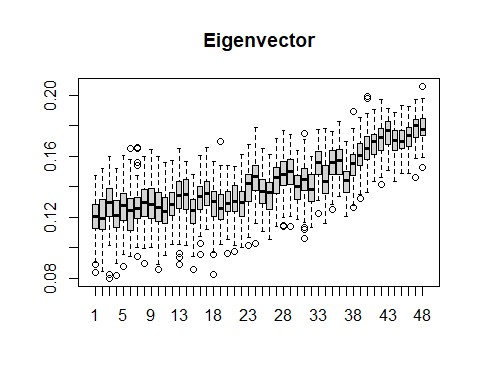
|  |  |
| --- | --- |
| bw.mat <- matrix(NA, nrow(d\_data3BR), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(BRprob3[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original similarity matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data3BR)[, 2]), ]), main = "Betweennes s")  ## Warning in as.tnet(net, type = "weighted one-mode tnet"): There were selfloops | |
| ## in the edgelist, these were removed |  |



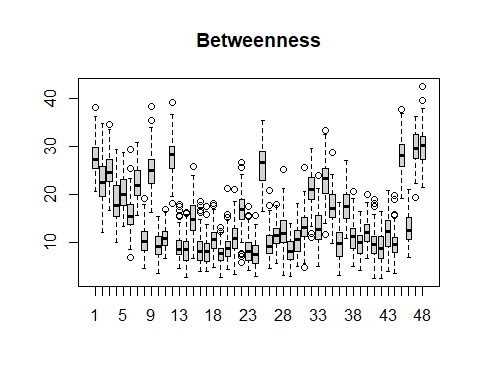
|  |
| --- |
| *# Run the script on the X2 distance matrix*  Xprob <- net.prob(d\_data1X01)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data1X01), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(Xprob[[i]])  }  *# show boxplot of degree centrality sorted by the degree cent score in the*  *# original distance matrix*  boxplot(t(dg.mat[order(rowSums(d\_data1X01)), ]), main = "Degree") |



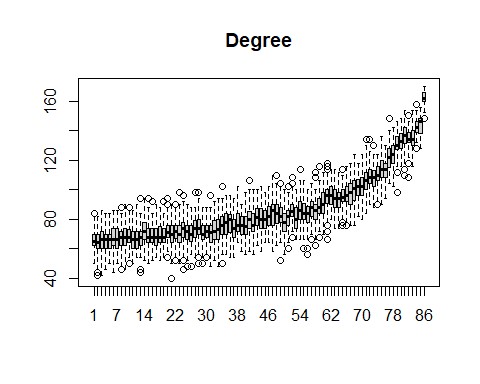
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data1X01), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(Xprob[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original distance matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data1X01)), ]), main = "Eigenvector") |



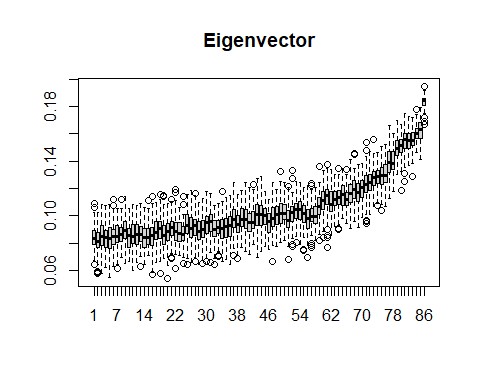
|  |
| --- |
| bw.mat <- matrix(NA, nrow(d\_data1X01), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(Xprob[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original distance matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data1X01)[, 2]), ]), main = "Betweenne ss") |



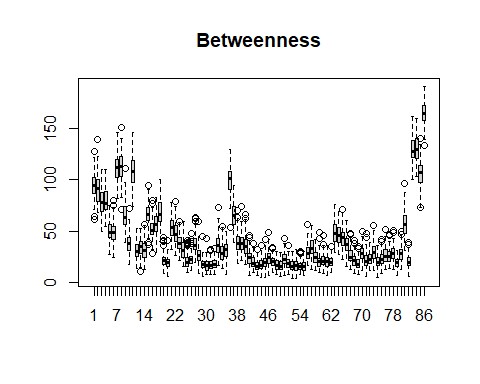
|  |
| --- |
| *# Run the script on the X2 distance matrix*  Xprob2 <- net.prob(d\_data2X01)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data2X01), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(Xprob2[[i]])  }  *# show boxplot of degree centrality sorted by the degree cent score in the original distance matrix*  boxplot(t(dg.mat[order(rowSums(d\_data2X01)), ]), main = "Degree") |



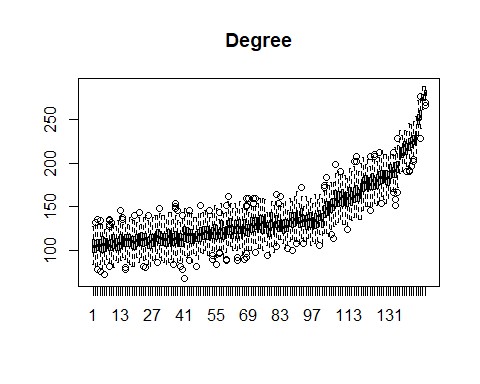
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data2X01), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(Xprob2[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original distance matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data2X01)), ]), main = "Eigenvector") |



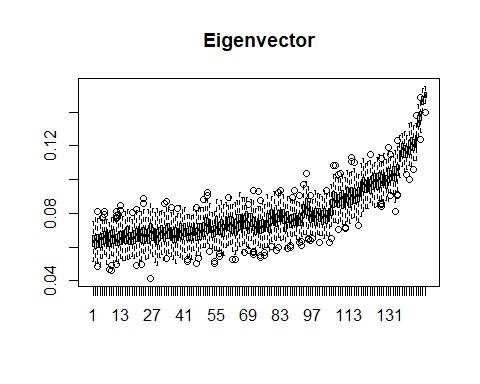
|  |
| --- |
| bw.mat <- matrix(NA, nrow(d\_data2X01), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(Xprob2[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original distance matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data2X01)[, 2]), ]), main = "Betweenne ss") |



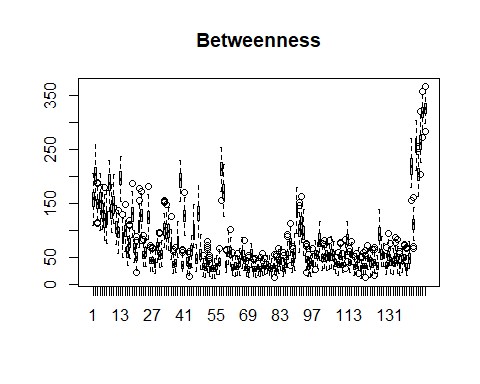
|  |
| --- |
| *# Run the script on the X2 distance matrix*  Xprob3 <- net.prob(d\_data3X01)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data3X01), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(Xprob3[[i]])  }  *# show boxplot of degree centrality sorted by the degree cent score in the*  *# original distance matrix*  boxplot(t(dg.mat[order(rowSums(d\_data3X01)), ]), main = "Degree") |



|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data3X01), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(Xprob3[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original distance matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data3X01)), ]), main = "Eigenvector") |



|  |
| --- |
| bw.mat <- matrix(NA, nrow(d\_data3X01), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(Xprob3[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original distance matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data3X01)[, 2]), ]), main = "Betweenne ss") |



## EVALUATE NETWORK DISTANCE CHANGES OVER TIME

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| library(igraph) | | | | | | | | | | |
| library(tidygraph) | | | | |  | | | | | |
|  | | | | |
| #load each dataset as an igraph network object | | | | | |  | | | | |
|  | | | | | |
| early <- graph\_from\_data\_frame(d\_data1, directed = F) | | | | | | | |  | | |
|  | | | | | | | |
| middle <- graph\_from\_data\_frame(d\_data2, directed = F) | | | | | | | | |  | |
|  | | | | | | | | |
| late <- graph\_from\_data\_frame(d\_data3, directed = F) | | | | | | |  | |
|  | | | | | | |
| #Calculate the largest distance between nodes at each time period | | | | | | | | | |  |
|  | | | | | | | | | |
| diameter(early) | | |  | | | | | | |
|  | | |
| ## [6] |  | |
|  |
| diameter(middle) | | | |  | | | | | |
|  | | | |
| ## [4] |  | | |
|  |
| diameter(late) | |  | |
|  | |
| ## [4] |  |