**Ceramic Analysis for “Evidence for extensive social networks as risk-mitigation strategies on Southwest Madagascar”**

Dylan S. Davis

9/16/2022

# Introduction

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 Southwest Madagascar following the protocol developed by Matt Peeples (2017). The original Script by Matt Peeples can be found at: [*http://www.mattpeeples.net/netstats.html*](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: "C:\Users\dylan\Documents\School\_Work\Dissertation\Ceramic\_Analysi s\Ceramic\_Network\_Data\Velondriake\_AOI.shp", layer: "Velondriake\_AOI" |

## with 1 features

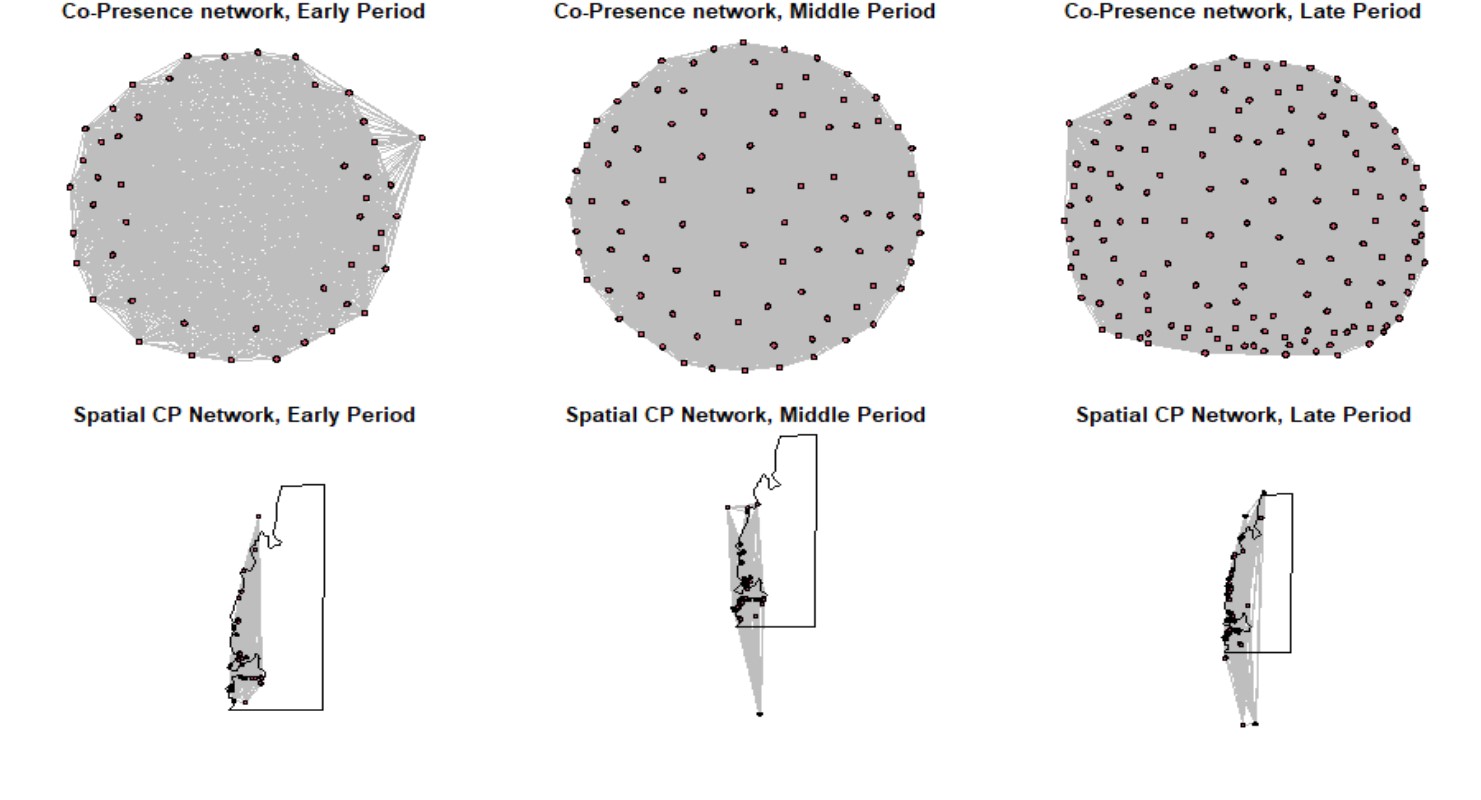
## It has 1 fields

##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

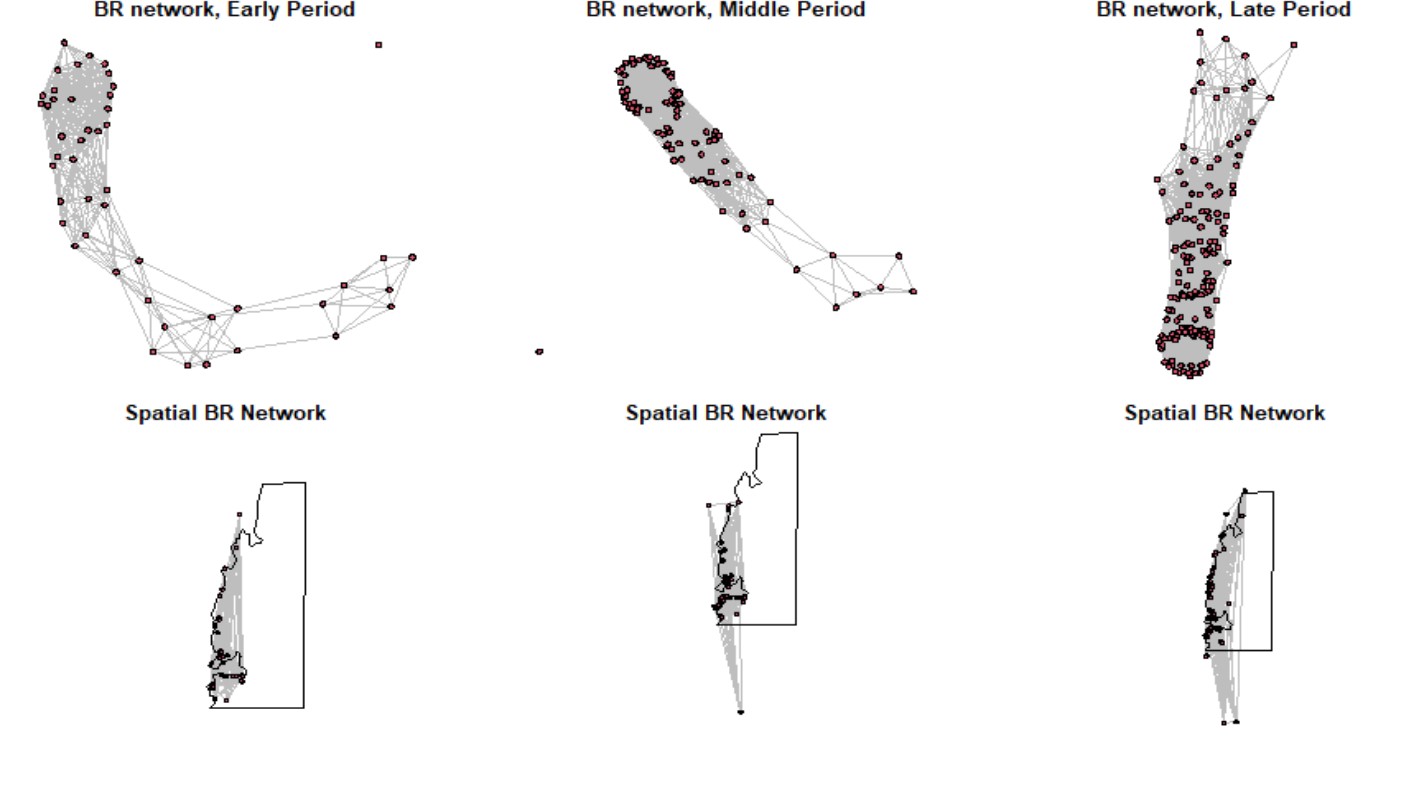
),

directed = F)

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

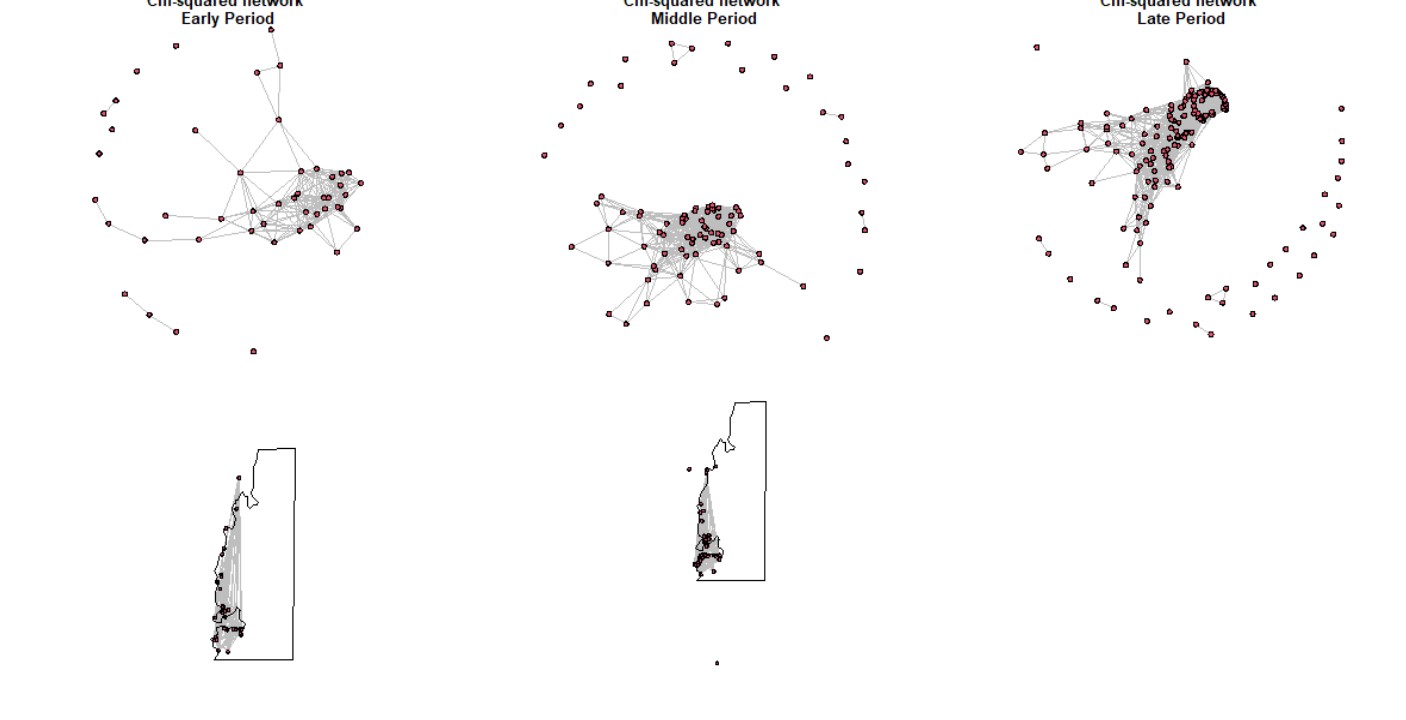
|  |
| --- |
| ),  directed = F)  BRnet\_d3 <- network(event2dichot(d\_data3BR, method = "absolute", thresh = 0.6  ),  directed = F)  *# Add names for nodes based on the row names of the original matrix*  BRnet\_d1 %v% "vertex.names" <- row.names(d\_data1BR)  BRnet\_d2 %v% "vertex.names" <- row.names(d\_data2BR)  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*

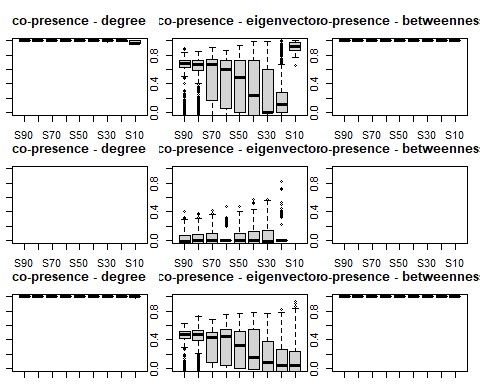
dg <- as.matrix(sna::degree(y, gmode = "graph"))

|  |
| --- |
| *# 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 |

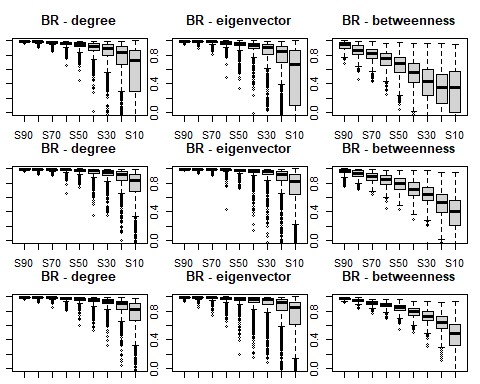
|  |
| --- |
| centralization(Pnet\_d2,betweenness,normalize=T) #calculate binary betweenness centralization  ## centralization(Pnet\_d2,betweenness,normalize=T) #calculate binary betweenn ess centralization  ## [1] 0    centralization(Pnet\_d3,evcent,normalize=T) #calculate binary eigenvector cent ralization  ## 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 |

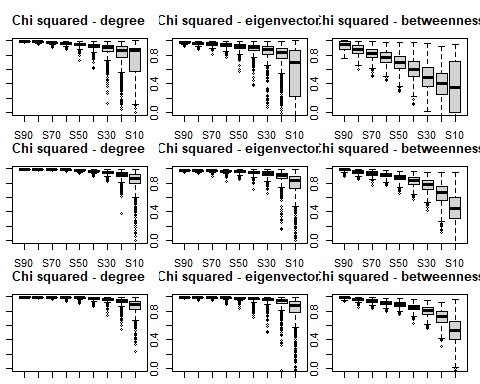
|  |  |
| --- | --- |
| out.list[[3]] <- bw.mat return(out.list)  } *# return the resulting list*  cop.rs\_d <- cv.resamp.bin(Pnet\_d1)cop.rs\_d2 <- cv.resamp.bin(Pnet\_d2)cop.rs\_d3 <- cv.resamp.bin(Pnet\_d3)BR.rs\_d <- cv.resamp.bin(BRnet\_d1)  BR.rs\_d2 <- cv.resamp.bin(BRnet\_d2)  BR.rs\_d3 <- cv.resamp.bin(BRnet\_d3)  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") |  |

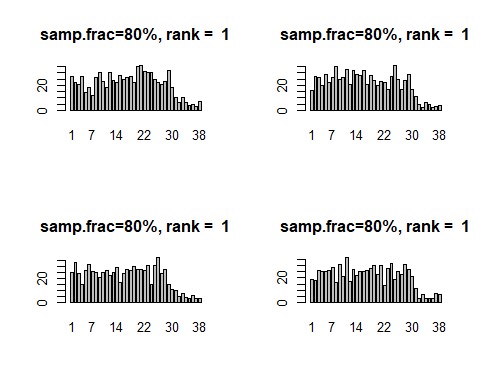


|  |  |
| --- | --- |
| par(mfrow = c(3, 3)) *# set up for 3 by 3 plotting* par(mar=c(1,1,3,1))    boxplot(X.rs\_d[[1]], ylim = c(0, 1), main = "Chi squared - degree", xlab = "s ampling fraction",  ylab = "Spearmans rho")  boxplot(X.rs\_d[[2]], ylim = c(0, 1), main = "Chi squared - eigenvector", xlab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

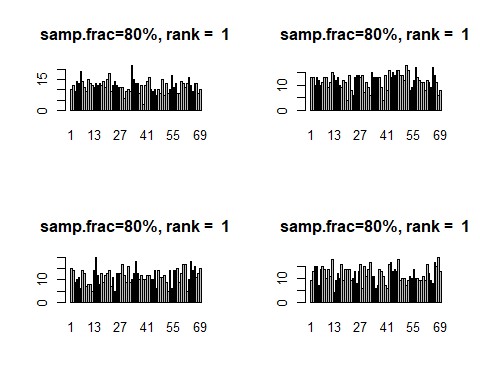


##Assess missing nodes within dataset

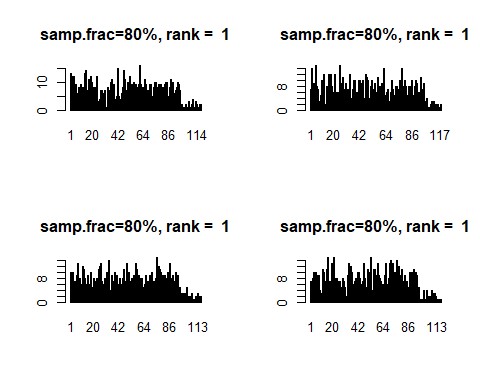
|  |
| --- |
| *#NODE ASSESSMENT*  nsim <- 1000 *#set number of replicates*    resamp.node <- **function**(x, samp.frac) { mat <- as.matrix(x) dim.x <- dim(mat)[1]  out.mat <- matrix(NA, dim.x, nsim) **for** (i **in** 1:nsim) {  sub.samp <- sample(seq(1, dim.x), size = round(dim.x \* samp.frac, 0),  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]]))} |



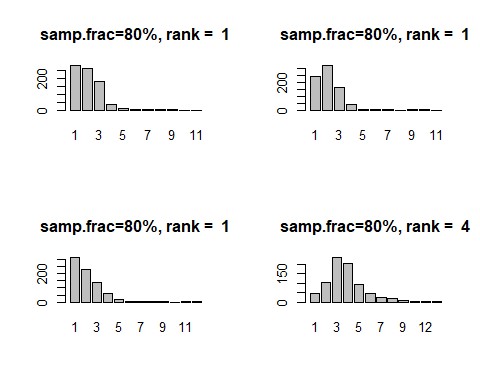
|  |
| --- |
| top.dg <- rank(-sna::degree(Pnet\_d2), ties.method = "min")par(mfrow = c(2, 2))  P.resamp <- resamp.node(Pnet\_d2, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots*  *# showing rank across all replicates* **for** (i **in**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



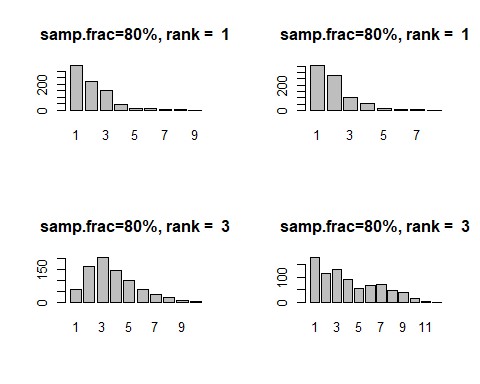
|  |
| --- |
| top.dg <- rank(-sna::degree(Pnet\_d3), ties.method = "min")par(mfrow = c(2, 2))  P.resamp <- resamp.node(Pnet\_d3, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4 as barplots # showing rank across all replicates*    **for** (i **in**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



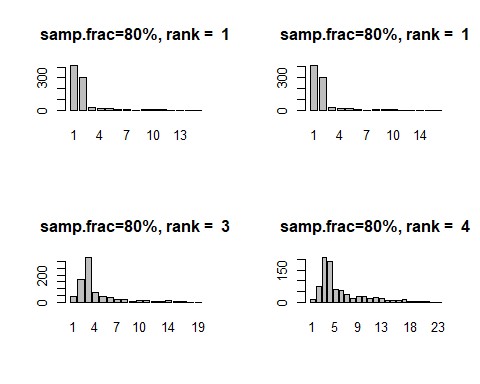
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(BRnet\_d1), ties.method = "min")  par(mfrow = c(2, 2))  BR.resamp <- resamp.node(BRnet\_d1, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top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



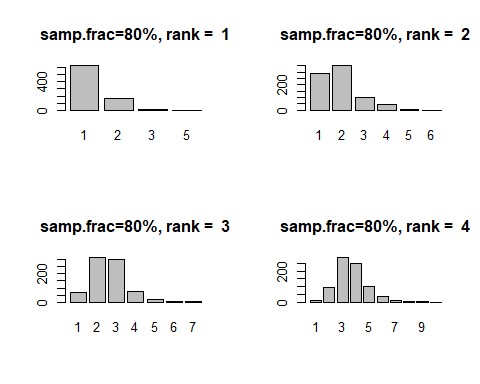
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(BRnet\_d2), ties.method = "min")  par(mfrow = c(2, 2))  BR.resamp <- resamp.node(BRnet\_d2, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top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



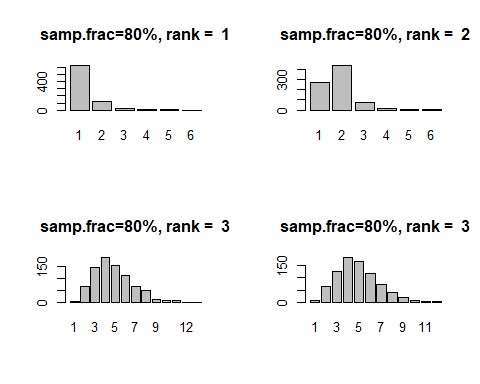
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(BRnet\_d3), ties.method = "min")  par(mfrow = c(2, 2))  BR.resamp <- resamp.node(BRnet\_d3, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top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



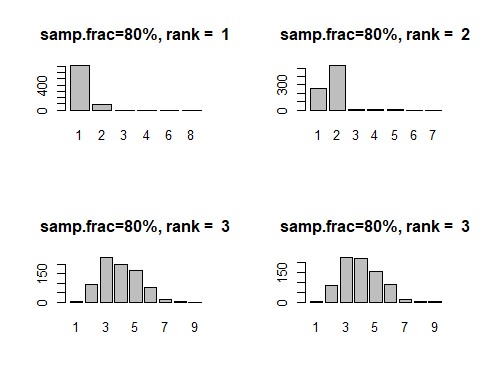
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(Xnet\_d1), ties.method = "min")par(mfrow = c(2, 2))  X.resamp <- resamp.node(Xnet\_d1, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4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



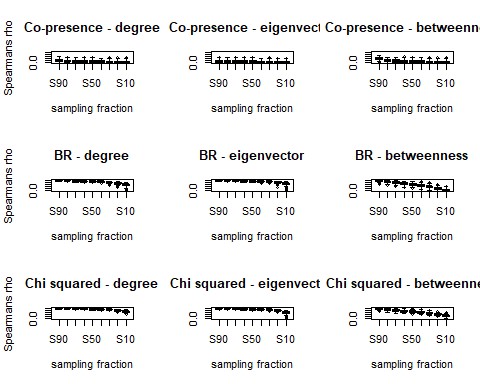
|  |
| --- |
| *# calculate the rank order of degree centrality in the BR network* top.dg <- rank(-sna::degree(Xnet\_d3), ties.method = "min")par(mfrow = c(2, 2))  X.resamp <- resamp.node(Xnet\_d3, samp.frac = 0.8) *#samp.frac is 80%*  *# calculate the rank order of the replicates and plot the top 4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



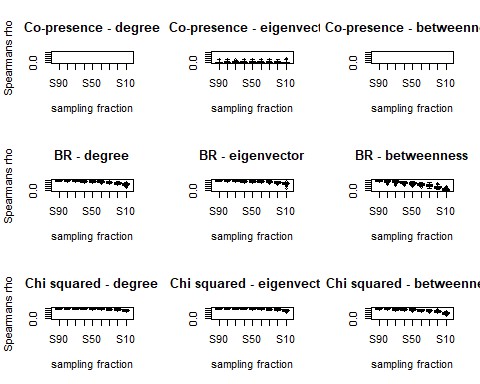
##Assess missing edges in dataset

|  |
| --- |
| *# 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) |

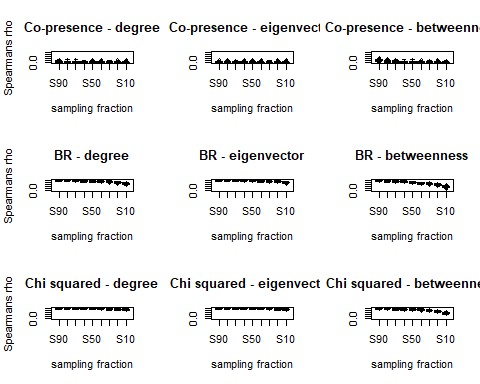
|  |
| --- |
| dg.mat[i, j] <- cor(temp.stats[, 1], stats.g[, 1], method = "spearman") ev.mat[i, j] <- cor(temp.stats[, 2], stats.g[, 2], method = "spearman") bw.mat[i, j] <- cor(temp.stats[, 3], stats.g[, 3], method = "spearman")  } }  out.list <- list() out.list[[1]] <- dg.mat out.list[[2]] <- ev.mat 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") |

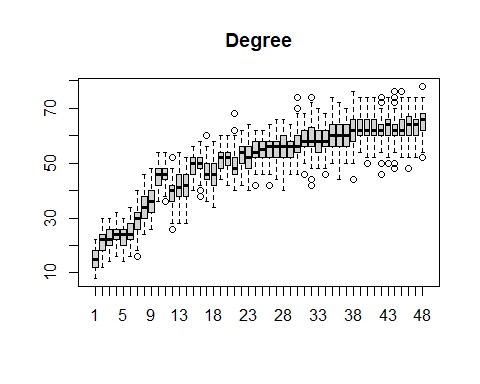


|  |
| --- |
| par(mfrow = c(3, 3))  boxplot(cop.edge.d3[[1]], ylim = c(0, 1), main = "Co-presence - degree", xlab  = "sampling fraction",  ylab = "Spearmans rho")  boxplot(cop.edge.d3[[2]], ylim = c(0, 1), main = "Co-presence - eigenvector", xlab = "sampling fraction")  boxplot(cop.edge.d3[[3]], ylim = 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") |

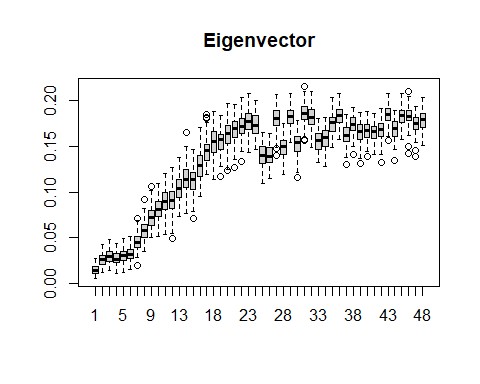


##Check for additional potential errors with missing data. This helps to validate our conclusions despite limited datasets

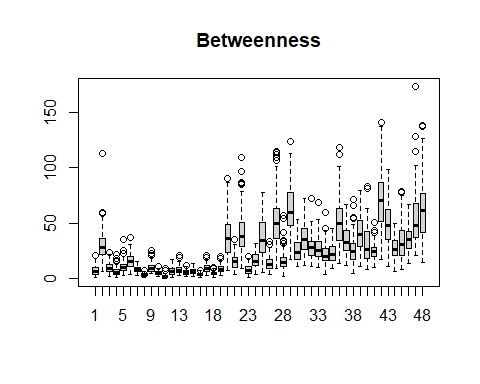
|  |
| --- |
| *###CHECK FOR SAMPLING ERRORS AND BIASES IN DATASET*  *## This code was originally written by Peeples (2017) and uses**## simulations to check for sampling issues*  nsim <- 100 *# we use 100 simulations due to computational limitations. We did assess the data using larger numbers of simulations (200, 500, 1000) and it p rovided identical results, but also occasionally crashed the computer used fo r 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") |



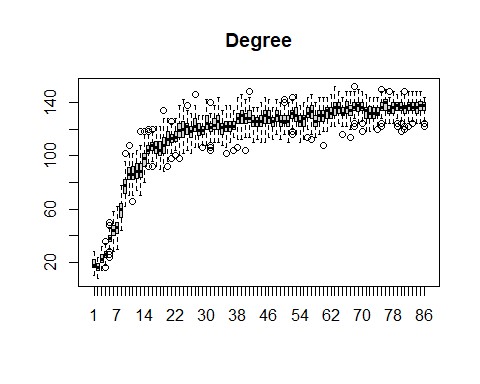
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data1BR), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(BRprob[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original similarity matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data1BR)), ]), main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



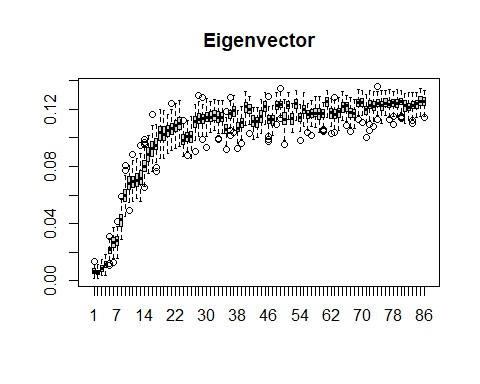
|  |  |
| --- | --- |
| bw.mat <- matrix(NA, nrow(d\_data1BR), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(BRprob[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original similarity matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data1BR)[,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



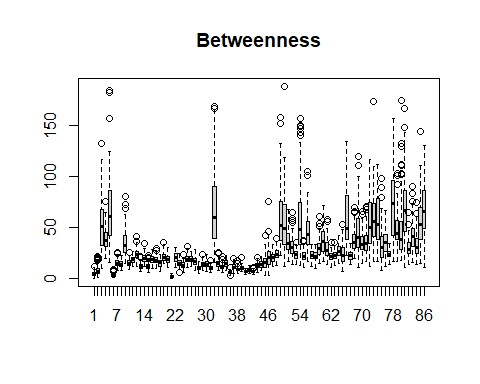
|  |
| --- |
| *# Run the script on the BR similarity matrix*  BRprob2 <- net.prob(d\_data2BR)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data2BR), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(BRprob2[[i]])  }  *# show boxplot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



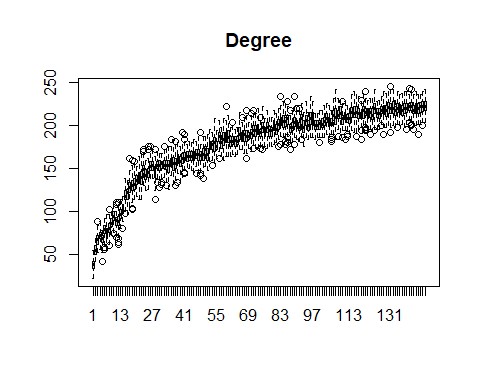
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data2BR), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(BRprob2[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original similarity matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data2BR)), ]), main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



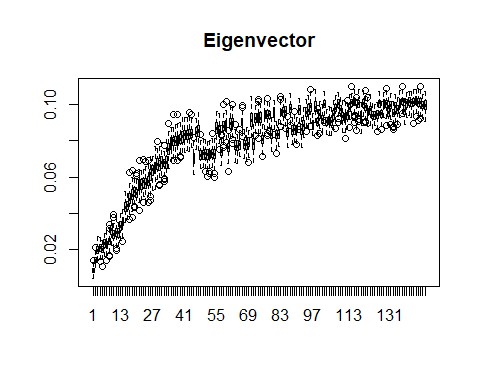
|  |  |
| --- | --- |
| bw.mat <- matrix(NA, nrow(d\_data2BR), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(BRprob2[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original similarity matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data2BR)[,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



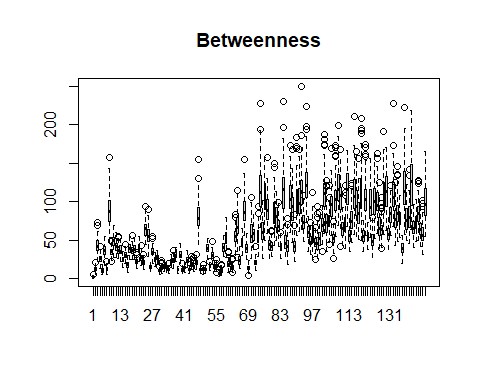
|  |
| --- |
| *# Run the script on the BR similarity matrix*  BRprob3 <- net.prob(d\_data3BR)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data3BR), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(BRprob3[[i]])  }  *# show boxplot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



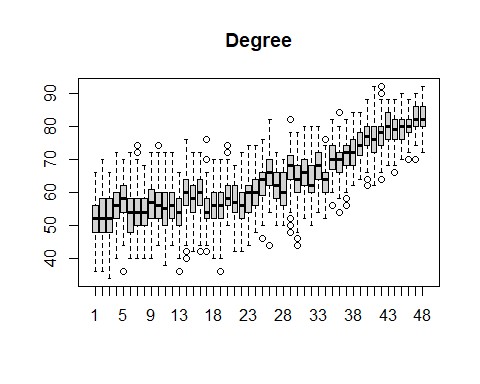
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data3BR), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(BRprob3[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original similarity matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data3BR)), ]), main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



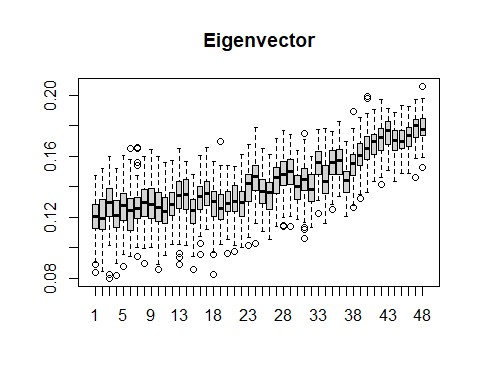
|  |  |
| --- | --- |
| bw.mat <- matrix(NA, nrow(d\_data3BR), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(BRprob3[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original similarity matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data3BR)[,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



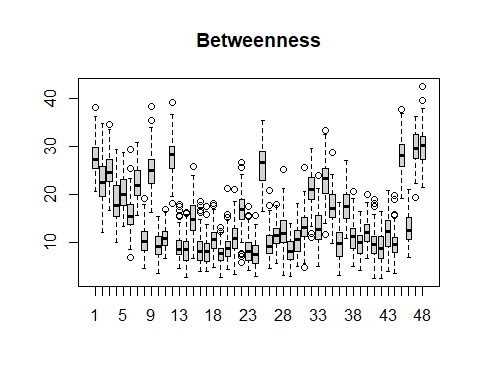
|  |
| --- |
| *# Run the script on the X2 distance matrix*  Xprob <- net.prob(d\_data1X01)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data1X01), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(Xprob[[i]])  }  *# show boxplot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



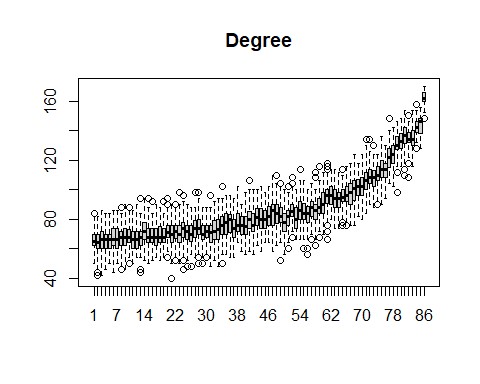
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data1X01), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(Xprob[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original distance matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data1X01)), ]), main =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



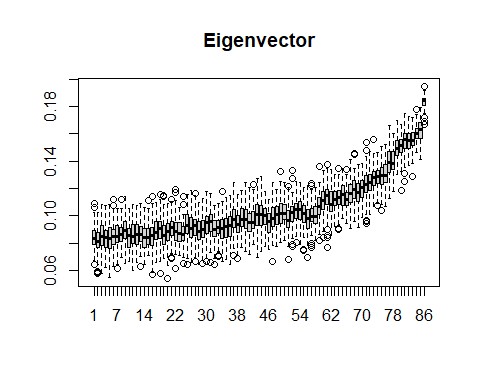
|  |
| --- |
| bw.mat <- matrix(NA, nrow(d\_data1X01), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(Xprob[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original distance matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data1X01)[,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



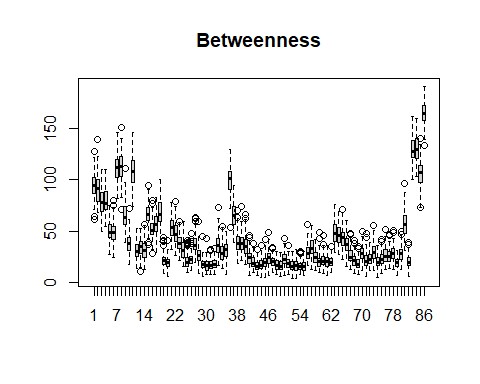
|  |
| --- |
| *# Run the script on the X2 distance matrix*  Xprob2 <- net.prob(d\_data2X01)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data2X01), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(Xprob2[[i]])  }  *# show boxplot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



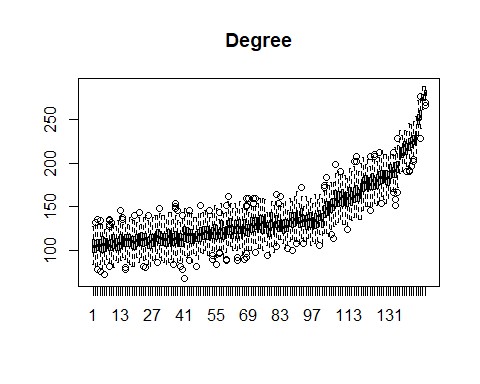
|  |
| --- |
| ev.mat <- matrix(NA, nrow(d\_data2X01), nsim)**for** (i **in** 1:nsim) {  ev.mat[, i] <- sna::evcent(Xprob2[[i]])  }  *# show boxplot of eigenvector centrality sorted by the EV cent score in the*  *# original distance matrix*  boxplot(t(ev.mat[order(sna::evcent(d\_data2X01)), ]), main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



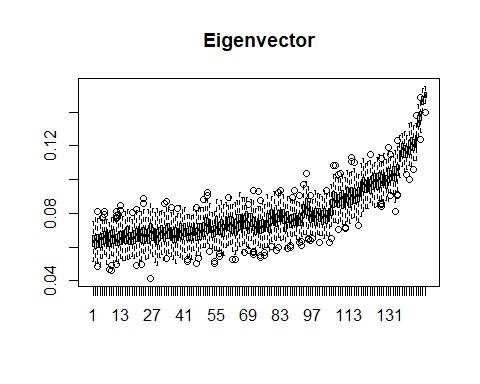
|  |
| --- |
| bw.mat <- matrix(NA, nrow(d\_data2X01), nsim)**for** (i **in** 1:nsim) {  bw.mat[, i] <- sna::betweenness(Xprob2[[i]])  }  *# show boxplot of betweenness centrality sorted by the betweenness cent*  *# score in the original distance matrix*  boxplot(t(bw.mat[order(betweenness\_w(d\_data2X01)[,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



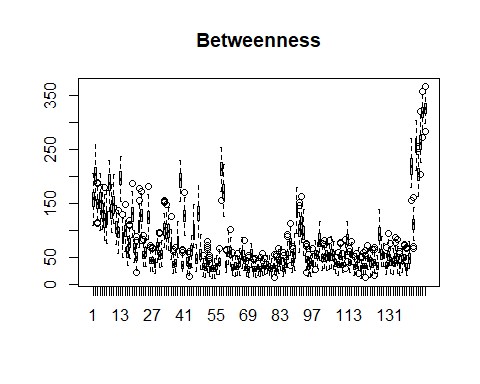
|  |
| --- |
| *# Run the script on the X2 distance matrix*  Xprob3 <- net.prob(d\_data3X01)  *# set up matrix and calculate eigenvector centrality for every replicate* dg.mat <- matrix(NA, nrow(d\_data3X01), nsim)**for** (i **in** 1:nsim) {  dg.mat[, i] <- sna::degree(Xprob3[[i]])  }  *# show boxplot 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] |  |