```
## Code to generate networks, and calculate Rcom from Cascadia PSeudorca data ##
## Written by AEH, based on code from Shizuka's online network tutorials #
# and from Shizuka & Farine (2015)##

## Data must be in a GBI format, with all restrictions already applied ##
## rows are individuals, and columns are groups ##
## ensure that all group codes are correctly assigned!!! ##
## You can build a GBI matrix in excel, using a pivot table ##

# load packages #
library(igraph)
library(bipartite)
library(asnipe)
library(assortnet)
library(ggplot2)
library(ggmap)
library(rnetcarto)
library(ecodist)
library(igraphdata)
library(statnet)
library(RColorBrewer)
library(tidyverse)
library(ggrepel)

# check working directory - this will change depending on your working directory location #
getwd()

# load in csv of data, in GBI format #
# columns represent groups, and rows represent individuals #
```

```
# I built this matrix in excel, using a pivot table #
data <- read.csv("Pseudorca_PQ2Dist2Seen5_19992021.csv", header=T, row.names = 1)

# replace all NAs with 0's, and all numbers >1 with 1's #
# this is necessary, since the matrix was built in excel #
data[is.na(data)] <- 0
data[data>1] <- 1

# transpose data matrix for asnipe #
# the matrix now has individuals as columns, and groups as rows #
gbi=t(data)

# convert data matrix into adjacency matrix #
adj.m=get_network(t(data), association_index="HWI")

# make a graph #
assoc.g=graph_from_adjacency_matrix(adj.m, "undirected", weighted=T)

# plot graph to check that it makes sense #
# set.seed allows for reproducible graphs #
set.seed(10)
layout1 <- layout_with_fr(assoc.g)
layout1 <- norm_coords(layout1, ymin=-1, ymax=1, xmin=-1, xmax=1)
# I usually have to run this step twice to make the layout fit the plot window #
plot(assoc.g, edge.width=E(assoc.g)$weight*10,
    layout=layout1*1.5,
    vertex.label="", vertex.size=8,
    rescale=F)
```

```
# let's get some network metrics #
n=vcount(assoc.g) # number of nodes - 174 #
m=ecount(assoc.g) # number of edges - 4297 #
dyads=n*(n-1)/2 # number of dyads - 15,051 #
components(adj.m) # components - 1 #

# let's try filtering out the HWIs < 0.3 #
adj.m2=get_network(t(data), association_index="HWI")
adj.m2[adj.m2 < 0.3] <- 0 # remove all HWIs < 0.3
assoc.g2=graph_from_adjacency_matrix(adj.m2, "undirected", weighted=T)
n2=vcount(assoc.g2) # number of nodes - 174 #
m2=ecount(assoc.g2) # number of edges - 1103 #
dyads=n2*(n2-1)/2 # number of dyads - 15,051 #
components(adj.m2) # number of components - 4 #
# plot with previous layout preserved to see what changes #
plot(assoc.g2, edge.width=E(assoc.g)$weight*10,
    layout=layout1*1.5,
    vertex.label="", vertex.size=8,
    rescale=F)

# plot with independent layout to preserve everyone's sanity #
layout2 <- layout_with_fr(assoc.g2)
layout2 <- norm_coords(layout2, ymin=-1, ymax=1, xmin=-1, xmax=1)
plot(assoc.g2,
    layout=layout2*1.25,
    vertex.label="", vertex.size=8,
    rescale=F)
```

```
# let's take a look at community structure! #
# We're going to start with a modularity-based approach, leading.eigenvector.community() #
# This approach is outlined in Newman 2006 #
lec = leading.eigenvector.community(assoc.g)
lec
# let's get some network metrics #
length(lec) # number of clusters - 5 #
modularity(lec) # modularity of lec method communities - 0.578 #
table(lec$membership) # let's check the number of individuals in each of our clusters #
# Cluster 1 - 66 #
# Cluster 2 - 37 #
# Cluster 3 - 7 #
# Cluster 4 - 6 #
# Cluster 5 - 58 #



# let's try edge.betweenness.community()#
eb = edge.betweenness.community(assoc.g)
eb
# let's get some network metrics #
length(eb) # number of clusters - 6 #
modularity(eb) # modularity of eb method communities - 0.581 #
table(eb$membership) # let's check the number of individuals in each of our clusters #
# Cluster 1 - 62 #
# Cluster 2 - 38 #
# Cluster 3 - 11 #
# Cluster 4 - 61 #
# Cluster 5 - 1 #
```

```
# Cluster 6 - 1 #



#let's try fastgreedy.community() #
fg= fastgreedy.community(assoc.g)
fg
length(fg) # number of clusters - 4 #
modularity(fg) # modularity of fg method communities - 0.605 #
table(fg$membership) # let's check the number of individuals in each of our clusters #
# Cluster 1 - 62 #
# Cluster 2 - 37 #
# Cluster 3 - 15 #
# Cluster 4 - 60 #

# let's try walktrap.community() #
wc = walktrap.community(assoc.g)
wc
length(wc) # number of clusters - 4 #
modularity(wc) # modularity of wc method communities - 0.605 #
table(wc$membership)
# Cluster 1 - 60 #
# Cluster 2 - 15 #
# Cluster 3 - 37 #
# Cluster 4 - 62 #



# let's try label.propagation.community #
lpc = label.propagation.community(assoc.g)
lpc
```

```
length(lpc) # number of clusters - 4 #
modularity(lpc) # modularity of lpc method communities - 0.600 #
table(lpc$membership)
# Cluster 1 - 64 #
# Cluster 2 - 37 #
# Cluster 3 - 14 #
# Cluster 4 - 59 #

# let's try cluster_louvain()#
cl = cluster_louvain(assoc.g)
cl
length(cl) # number of clusters - 4 #
modularity(cl) # modularity of cl method communities - 0.605 #
table(cl$membership)
# Cluster 1 - 62 #
# Cluster 2 - 37 #
# Cluster 3 - 15 #
# Cluster 4 - 60 #



# let's revisualize the network, to see what cluster assignments look like #
# can change colors however necessary for particular community assignment method #
# lec #
node.colors=membership(cl)
plot(assoc.g, edge.width=E(assoc.g)$weight*10,
    layout=layout1*1.5,
    vertex.label="", vertex.size=8, vertex.color=node.colors,
    rescale=F)
```

```
## another example with walktrap.community ##
node.colorswc=membership(wc)
plot(assoc.g, edge.width=E(assoc.g)$weight*10,
    layout=layout1*1.5,
    vertex.label="", vertex.size=8, vertex.color=node.colorswc,
    rescale=F)

## bring network into ggplot2 format ##
## Integrating ggplot and igraph ##
## includes code to save csvs of edgelist and node coordinates for future reproducible networks ##
# get node coordinates #
nodes <- as.data.frame(layout1)
nodes$names <- V(assoc.g)$name
nodes$wcmembership <- wc$membership
nodes$ebmembership <- eb$membership
nodes$lecmembership <- lec$membership
nodes$lpcmembership <- lpc$membership
nodes$fgmembership <- fg$membership
nodes$clmembership <- cl$membership
colnames(nodes) = c("X1","X2", "names", "wccluster", "ebcluster", "leccluster", "lpccluster", "fgcluster",
"clcluster")
write.csv(nodes, "Pc_Dist2PQ2_19992021_Seen5_fullnetwork_nodecoords_clustermembership.csv")
#get edges, which are pairs of node IDs #
edgelist <- get.data.frame(assoc.g)
# match coordinates to edgelist #
edgelist$from.x <- nodes$X1[match(edgelist$from, nodes$names)]  #  match the from locations from
the node data.frame we previously connected
edgelist$from.y <- nodes$X2[match(edgelist$from, nodes$names)]
edgelist$to.x <- nodes$X1[match(edgelist$to, nodes$names)]  #  match the to locations from the node
data.frame we previously connected
```

```
edgelist$to.y <- nodes$X2[match(edgelist$to, nodes$names)]
write.csv(edgelist, "Pc_Dist2PQ2_19992021_Seen5_fullnetwork_edgelist.csv")



# plot full networks with ggplot! From here you can customize the network however you want! #
# colors will have to be adjusted manually to keep clusters consistent #
# full network with wc #
ggplot() +
  geom_segment(data=edgelist, aes(x=from.x,xend = to.x, y=from.y,yend = to.y), color="grey", size=0.25)
+
  geom_point(data=nodes, aes(x=X1, y=X2, color=factor(wccluster)), shape=16, size=4) +
  theme_void() + scale_color_manual(values=c("#a6cee3", "#1f78b4", "#b2df8a", "#33a02c")) +
  theme(legend.position = "")
# full network with eb #
ggplot() +
  geom_segment(data=edgelist, aes(x=from.x,xend = to.x, y=from.y,yend = to.y), color="grey", size=0.25)
+
  geom_point(data=nodes, aes(x=X1, y=X2, color=factor(ebcluster)), shape=16, size=4) +
  theme_void() + scale_color_manual(values=c("#33a02c", "#b2df8a", "#1f78b4", "#a6cee3", "#000000",
"#ff91a4")) +
  theme(legend.position="")

## reproducible ggplot2 network with HWI's <0.3 filtered out #
nodesfiltered <- as.data.frame(layout2)
nodesfiltered$names <- V(assoc.g2)$name
nodesfiltered$wcmembership <- wc$membership
nodesfiltered$ebmembership <- eb$membership
nodesfiltered$lecmembership <- lec$membership
nodesfiltered$lpcmembership <- lpc$membership
nodesfiltered$fgmembership <- fg$membership
```

```
nodesfiltered$clmembership <- cl$membership

colnames(nodesfiltered) = c("X1","X2", "names", "wccluster", "ebcluster", "leccluster", "lpccluster", "fgcluster", "clcluster")

write.csv(nodesfiltered, "Pc_Dist2PQ2_19992021_Seen5_filterednetwork_nodecoords_clustermembership.csv")

#get edges, which are pairs of node IDs #

edgelistfiltered <- get.data.frame(assoc.g2)

# match coordinates to edgelist #

edgelistfiltered$from.x <- nodesfiltered$X1[match(edgelistfiltered$from, nodesfiltered$names)]  # match the from locations from the node data.frame we previously connected

edgelistfiltered$from.y <- nodesfiltered$X2[match(edgelistfiltered$from, nodesfiltered$names)]

edgelistfiltered$to.x <- nodesfiltered$X1[match(edgelistfiltered$to, nodesfiltered$names)]  #  match the to locations from the node data.frame we previously connected

edgelistfiltered$to.y <- nodesfiltered$X2[match(edgelistfiltered$to, nodesfiltered$names)]

write.csv(edgelistfiltered, "Pc_Dist2PQ2_19992021_Seen5_filterednetwork_edgelist.csv")


# plot HWIs <0.3 network (filtered) with ggplot! From here you can customize the network however you want! #

# colors will have to be adjusted manually to keep clusters consistent #

# walktrap with HWIs filtered #

ggplot() +

  geom_segment(data=edgelistfiltered, aes(x=from.x,xend = to.x, y=from.y,yend = to.y), color="grey", size=0.25) +

  geom_point(data=nodesfiltered, aes(x=X1, y=X2, color=factor(wccluster)), shape=16, size=4) +

  theme_void() + scale_color_manual(values=c("#a6cee3", "#1f78b4", "#b2df8a", "#33a02c")) +

  theme(legend.position="")

# EB with HWIs filtered #

ggplot() +

  geom_segment(data=edgelistfiltered, aes(x=from.x,xend = to.x, y=from.y,yend = to.y), color="grey", size=0.25) +

  geom_point(data=nodesfiltered, aes(x=X1, y=X2, color=factor(ebcluster)), shape=16, size=4) +
```

```
 theme_void() + scale_color_manual(values=c("#33a02c", "#b2df8a", "#1f78b4", "#a6cee3", "#000000",
"#ff91a4")) +
 theme(legend.position="")


ggsave("Pseudorca_Dist2+PQ2+1999-2021Seen5+_Walktrapclusters_filtered_withoutlegend.jpg",
dpi=600)


# base R plots with HWI's <0.3 filtered out #
# I didn't use these in the manuscript, but am keeping the code here for documentation #
# lec #
plot(assoc.g2,
    layout=layout2*1.5,
    vertex.label="",
    vertex.size=8,
    rescale=F, vertex.color=node.colors)
# eb #
node.colorseb = membership(eb)
plot(assoc.g2,
    layout=layout2*1.5,
    vertex.label="", vertex.size=8,
    rescale=F, vertex.color=node.colorseb)
```

```
# fg #
node.colorsfg = membership(fg)
plot(assoc.g2,
    layout=layout2*1.5,
    vertex.label="", vertex.size=8,
    rescale=F, vertex.color=node.colorsfg)
# wc #
node.colorswc = membership(wc)
plot(assoc.g2,
    layout=layout2*1.5,
    vertex.label="", vertex.size=8,
    rescale=F, vertex.color=node.colorswc)
# sc #
node.colorssc = membership(sc)
plot(assoc.g2,
    layout=layout2*1.5,
    vertex.label="", vertex.size=8,
    rescale=F, vertex.color=node.colorssc)
# lpc #
node.colorslpc = membership(lpc)
plot(assoc.g2,
    layout=layout2*1.5,
    vertex.label="", vertex.size=8,
    rescale=F, vertex.color=node.colorslpc)
# cl #
node.colorscl = membership(cl)
plot(assoc.g2,
    layout=layout2*1.5,
    vertex.label="", vertex.size=8,
```

```
    rescale=F, vertex.color=node.colorscl)
```

```
## RCOM CALCULATION - Code from Shizuka & Farine (2015) ##
calc_rc=function(data, n.bootstraps=100, plot.result=T, ncoms=T){

  # Create space to store results from bootstraps
  network.community <- matrix(0,ncol(data),ncol(data))
  network.present <- matrix(0,ncol(data),ncol(data))
  ncoms_vec <- numeric()

  # 1. Calculate network - results in nxn matrix of association weights for each dyad #
  network <- get_network(data,data_format="GBI", association_index="HWI")

  # 2. Calculate community membership of the observed network ##
  # may be able to replace fastgreedy.community with alternative methods from igraph ##
  community.observed <-
cluster_louvain(graph.adjacency(network,mode="undirected",weighted=TRUE))

  # 3. Main bootstrapping method: i) Bootstrap the observed data, ii) recalculate the network,
  #    iii) recalculate community membership, iv) check if both individuals are observed

  for (i in 1:n.bootstraps) {
    # This step bootrstraps the sampling periods
```

```
  gbi.boot <- data[sample(1:nrow(data),nrow(data),replace=TRUE),]
  network.boot <- get_network(gbi.boot,data_format="GBI", association_index="HWI")

  # This step calculates the community membership from the bootstrapped network
  community.boot <-
cluster_louvain(graph.adjacency(network.boot,mode="undirected",weighted=TRUE))

  # This step adds 1 to any dyads in the same community
  network.community <- network.community + outer(community.boot$membership,
community.boot$membership,"==")

  # This step adds 1 to any dyads that are both present (in this case if they have at least 1 edge)
  network.present <- network.present +
outer((rowSums(network.boot)>0),(rowSums(network.boot)>0),"*")

  # save number of communities from each bootstrap into a vector #
  ncoms_vec <- append(ncoms_vec, length(community.boot))
 }
 # End bootstrap

 # Calculate proportion of times observed in the same community
 P <- network.community/network.present
 P[!is.finite(P)] <- 0

 # Calculate assortment from known community membership
 rc <- assortment.discrete(P,community.observed$membership)$r
```

```
  #if the argument plot.result=T, then generate plot network of probabilities that nodes are assigned to the same community in bootstraps. It will be saved as pdf file called "rc_result.pdf" in your R output folder

  if(plot.result) {

    pdf("rc_result.pdf")

    diag(P)=0

    g <<- graph.adjacency(P, "undirected", weighted=TRUE)

    plot(g, edge.width=E(g)$weight, vertex.label="", vertex.size=5, vertex.color=membership(community.observed))

    dev.off()

  }


  if(ncoms){write.csv(ncoms_vec, "bootstrapped_ncoms.csv")}


  return(rc)

}



## code to actually run the bootstraps - make sure that data = gbi!!! ##

calc_rc(gbi, n.bootstraps=1000, plot.result=T, ncoms=T)



## extract probability graph csvs for replicable results in ggplot2 ##

# set node coordinates #

set.seed(10)

layout3 <- layout_with_fr(g)

layout3 <- norm_coords(layout3, ymin=-1, ymax=1, xmin=-1, xmax=1)

# check that plot matches with function PDF output #

plot(g, edge.width=E(g)$weight*10,

     layout=layout3*1.5,
```

```
    vertex.label="", vertex.size=8,
    rescale=F)
# build csv of probability graph nodes #
probabilitynodes <- as.data.frame(layout3)
probabilitynodes$names <- V(g)$name
probabilitynodes$membership <- cl$membership
colnames(probabilitynodes) = c("X1","X2", "names", "membership")
write.csv(probabilitynodes, "Pc_Dist2PQ2_19992021_Seen5_CL_probabilitynodes.csv")
#get edges, which are pairs of node IDs #
probabilityedgelist <- get.data.frame(g)
# match coordinates to edgelist #
probabilityedgelist$from.x <- probabilitynodes$X1[match(probabilityedgelist$from,
probabilitynodes$names)]  #  match the from locations from the node data.frame we previously
connected
probabilityedgelist$from.y <- probabilitynodes$X2[match(probabilityedgelist$from,
probabilitynodes$names)]
probabilityedgelist$to.x <- probabilitynodes$X1[match(probabilityedgelist$to,
probabilitynodes$names)]  #  match the to locations from the node data.frame we previously connected
probabilityedgelist$to.y <- probabilitynodes$X2[match(probabilityedgelist$to,
probabilitynodes$names)]
write.csv(probabilityedgelist, "Pc_Dist2PQ2_19992021_Seen5_CL_probabilityedgelist.csv")
# check that probability graph makes sense #
ggplot() +
  geom_segment(data=probabilityedgelist, aes(x=from.x,xend = to.x, y=from.y,yend = to.y), color="grey",
size=0.25) +
  geom_point(data=probabilitynodes, aes(x=X1, y=X2, color=factor(membership)), shape=16, size=4) +
  theme_void() +
  theme(legend.position = "")
```

```
## Plot histogram of bootstrapped outputs ##


bootstrappedncoms <- read.csv("bootstrapped_ncoms.csv")
colnames(bootstrappedncoms)
colnames(bootstrappedncoms) <- c("bootstrap", "ncoms")
ggplot() + geom_histogram(data=bootstrappedncoms, aes(x=ncoms))
```