Network Structure

Kyra Bankhead

2023-03-02

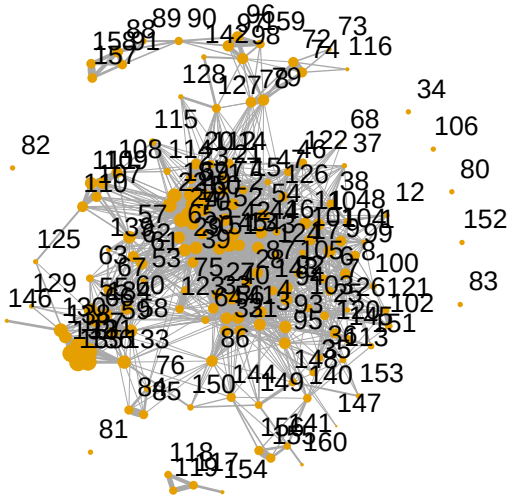
In this markdown I will:

1. Create the network structure from the association matrix.
2. Evaluate local and global network metrics.
3. Permutate the link weights using the WalkTrap algorithm.
4. Evaluate modularity.

PART 1: *Network Structure*

```
## load all necessary packages
require(igraph) # Look at Dai Shizuka/Jordi Bascompte
require(tnet) # For weights
require(sna)
require(statnet)

# Read in social association matrix
setwd("C:/Users/bankh/My_Repos/Dolphins/data")
# Read in social association matrix
nxn <- readRDS("nxn.RData")

# Test one year at a time
year <- 1
nxn <- nxn[[year]]

## Create social network
ig <- graph_from_adjacency_matrix(as.matrix(nxn),
                                  mode = c("undirected"),
                                  weighted = TRUE,
                                  diag = F, # No loops
                                  add.colnames = T,
                                  add.rownames = NA)

# Plot network
plot(ig,
     layout = layout_with_fr(ig),
     # link weight, rescaled for better visualization
     edge.width= E(ig)$weight*4,
```

```
    # node size as degree (rescaled)
    vertex.size= sqrt(igraph::strength(ig, vids = V(ig), mode = c("all"), loops = TRUE) *10 ),
    vertex.frame.color= NA, #"black",
    vertex.label.family = "Helvetica",
    vertex.label.color="black",
    vertex.label.cex=0.8,
    vertex.label.dist=2,
    # edge.curved=0,
    vertex.frame.width=0.01,
)
```



PART 2: *Network Metrics*

Local Network Metrics

- Local clustering coefficient: Measure of the prevalence of node clusters in a network.
- Betweeness: A high betweenness means that the individual is in the communication path of other individuals, therefore, the individuals it interacts with, depend on its presence.
- Closeness: The larger the closeness centrality is for an individual, the more rapidly and easily it can influence the behavior of others.
- Degree: # Individual's associates
- Strength: Total strength of an individuals' associations

Global Network Metrics

- Size: Number of nodes.
- Density/Connectance: Proportion of realized links (observed/possible links).
- Average Path Length (geodesic): Measures the shortest distance between two random nodes then average shortest pathways between all pairs of nodes. Shows how far apart any pair of individuals will be on average.
- Geodesic path: the shortest path through the network from one node to another (l).
- Diameter: Length of the longest geodesic path (d).
- Clustering coefficient: Tendency of nodes to cluster in the network (Are the friends' friends also friends?).

PART 3: *Permutate Link Weights*

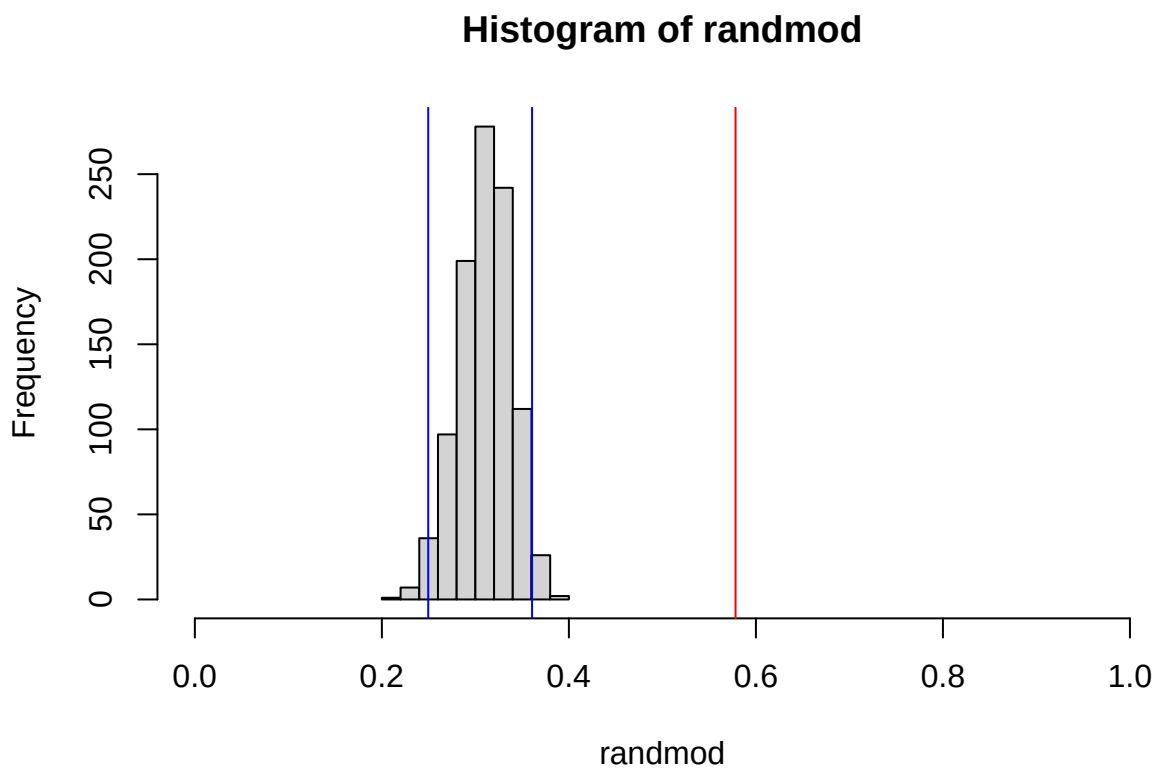
Walktrap algorithm breakdown with one interation

Permutate with multiple interations

```
# Run modularity permutations 1000 times
iter = 1000
randmod = numeric()
for(i in 1:iter){
  # Save the edgelist into a new object
  auxrand <- el
  # igraph format
  igrand <- graph.edgelist(auxrand[,1:2]) # Create a network from the list of nodes
  E(igrand)$weight <- auxrand[,3] # Add link weights
  igrand <- as.undirected(igrand) # Make undirected graph
  # Permutate the link weights
  E(igrand)$weight <- sample(E(igrand)$weight)
  # calculate the modularity Q-value
  rand_walk <- walktrap.community(igrand)
  randmod[i] <- modularity(rand_walk) # Save Q-value into a vector
}


## Calculate the 95% confidence interval (two-tailed test)
ci = quantile(randmod, probs=c(0.025, 0.975), type=2)

## Compare with the empirical Q-value
data.frame(Q=modularity(dolphin_walk), LowCI=ci[1], HighCI=ci[2])
```

```
##              Q     LowCI    HighCI
## 2.5% 0.5782209 0.2496046 0.3606259
```

```
## Visualization random Q distribution
hist(randmod, xlim=c(0,1))
### Empirical Q-value
abline(v= modularity(dolphin_walk), col="red")
### 2.5% CI
abline(v= ci[1], col="blue")
### 97.5% CI
abline(v= ci[2], col="blue")
```



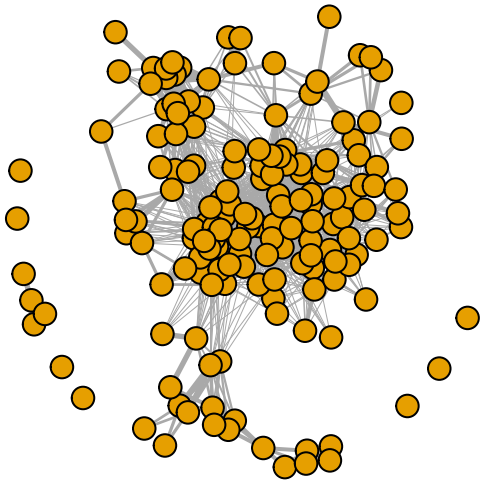
We can reject the null hypothesis that individuals cluster at random and conclude that there is evidence that modularity is higher than what we would expect by chance.

PART 4: *Modularity*

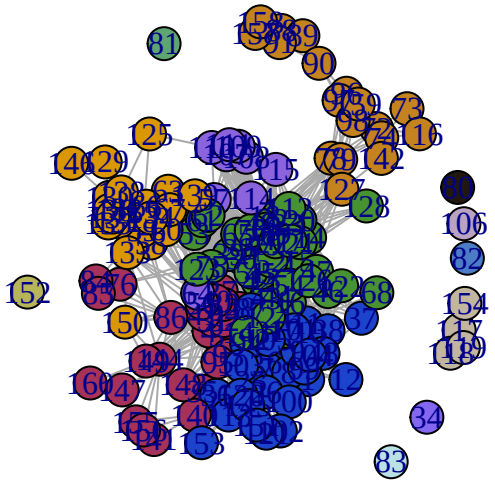
- Newman's Q modularity: Stopping parameter Q removes links according to the betweenness.

```
# Create a network from the first two columns
dolp_ig <- graph.edgelist(el[,1:2])
# Add the edge weights to this network by assigning an edge attribute called 'weight'.
E(dolp_ig)$weight <- as.numeric(el[,3])
# Create undirect network
dolp_ig <- as.undirected(dolp_ig)
```

```r
# Plot
plot(dolp_ig, edge.width=E(dolp_ig)$weight*4, vertex.size=10, vertex.label=NA, edge.curved=F)
```



```r
# Newman's Q modularity
newman <- cluster_leading_eigen(dolp_ig, steps = -1, weights = E(dolp_ig)$weight,
                                start = NULL, options = arpack_defaults, callback = NULL,
                                extra = NULL, env = parent.frame())

# Assign a random color to individuals of each module ('modules')
# Random color scheme
col <- rgb(runif(14), runif(14), runif(14))

V(dolp_ig)$color <- NA
for (i in 1:max(newman$membership)){
  sample(col)
  V(dolp_ig)$color[which(newman$membership==i)] = col[i]
}

plot(dolp_ig)
```



Since these modules can represent functional units, I need to test which mechanisms drive the modular topology by creating null models.