Collobration network analysis

Amir Imani

1 September 2016

Nowadays in every field, people collobrate with each other.There are alot of intersing collobation network analysis projects out there like rappers (<http://arxiv.org/pdf/physics/0511215.pdf>),co-stardom analysis (<https://en.wikipedia.org/wiki/Co-stardom_network>), and scientific collobarion. The colloration network is formed as indvidulas with different skills/knowledge/professions interact with each other to facilitate fulfillment of tasks to a mutally shared goal. Collboration provides opportunity to all the parties involved to discover new knowledge/experinece/skills and consequenty combine them to address a complex challenge.

Since for I already have a few thousands journal articles on ethics of Synthetic Biology written from 2004 to 2014, I decided to write this walkthrough on how to do the co-authorship analysis. Scientific collaborative networks are a hallmark of contemporary academic research. Scientists are no longer independent players, but members of scientific cooperation networks looking for solutions to social, political, economic and technological problems, which, usually, require multidisciplinary approaches (<http://goo.gl/GtwV02>). Scientific collaboration can also help broaden the scope of a research project and foster innovation as it provides access to different disciplines.

Co-authorship analysis enables us to understand and disvocer scientific collaboration patterns. It can be used to asses the extend of collaboration within a research area, evaluate the relation between authors and their significance in a given network, contirbution of insitutions or larger organisaitons, formation of research groups, and so on.

Regardless of what kind of collaboration network you want to analysis, you'll need an edge list where individulas are unique nodes with edges connecting them to each other if they have colloborated. In scientific co-authorship networks, nodes represent authors, organizations or countries, which are connected when they share the authorship of a paper.

## ANALYSIS

### importing the file

For this post, I picked a random sample of 100 articles and scraped the authors (which I will write about in another post) to start with. It looks like the table below. You can find all the files and codes on my ***github repo***.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| First Author |  |  |  |  |  |
| Adam Arkin | Pamela A. Silver | George Chruch |
| Ahmad S. Khalil | Agomoni Ganguli-Mitra |
| Ali Nouri | Alan Irwin |
| Amelie Cserer | Alexander van Oudenaarden |
| Anna Deplazes | Alexandra Seiringer |
| Baker David | Anna Labno | Collins Jim | Endy Drew | Jacobson Joseph | Keasling Jay | Modrich Paul |
| Barry Canton | Antoine Danchin | Drew Endy |
| Brian J Yeh | Arthur Caplan |
| Brianna Pearson | Barbara A. Fechta | Kyri Bye-Nagel | Scott Tonid | Laurie J Heyer | Malcolm Campbell |
| C. Verharen | C. Prowse | G. Middendorf | M. Castro-Sitiriche | G. Kadoda |
| Christian L Barrett | Chris J. Paddonb | Hyun Uk Kim | Bernhard Ë Palsson | Sang Yup Lee |
| Christina D. Smolke | Christopher F Chyba |

source\_file = read.csv(file.choose(), sep = ";", header = F)

### Converting a table to an edgle list

There are many approaches to turn a table to an edge-list. I defined a function using **reshape** library. The idea is to melt the table starting from the first column, then drop it and move to the next column.

library(reshape)  
  
Matrix2Edge <- function(x){  
 final <- data.frame("a"=character(),"b"=character()) #an empty dataframe to add the edge list to  
   
 for (i in 1:(ncol(x)-1)){  
 id = paste("V", i, sep="")  
 edge\_list <- melt(x, id = (id))  
 x[id] <- NULL  
 keeps <- c(id, "value")  
 edge\_list <- edge\_list[keeps]  
 colnames(edge\_list) <- colnames(final)  
 final <- rbind(final,edge\_list)  
 }  
 final <- final[!(final$b=="" | final$b==" "),]  
 return(final)  
}

Then Using the function with the > source\_file as the input

edge\_list <- Matrix2Edge(source\_file)

The result is a two-colmumn table where each row is an edge indiciating every two authors who collobareted in a paper. \* One important remark about scientific co-authorship is that not all authors have contributed the same.Based on the field of research, the first author and last author have different level of contirbution comparing to the others. This can be addressed by assgiging a weight factor to those authors.

### ENTITY RESOLUTION

The next step is to make sure that variation in authors names like misspelling, missing the middle name, or the order of first and last name have not created duplicates. This process is called enetity resolution and there are many approaches to address it in *machine learnin*.

But first, we need to clean up the text. In this case it includes removing numbers, special characters, triming white space, and converting all letters to lowercase, etc to minimize possible discrepencies. To do so, Im using **dplyr** library to convert all letters to lowercase.

library(dplyr)  
  
edge\_list <- as.data.frame(sapply(edge\_list, function(x) gsub("[^[:alnum:]]"," ", x))) #removes all non-alphanumerical characters (like punctuations, special characters, etc)  
  
edge\_list <- as.data.frame(sapply(edge\_list, function(x) trimws(x))) #removes leading and trainling whitespaces  
  
edge\_list <- mutate\_each(edge\_list, funs(tolower))

Now entity resolution time (write more about the distance function) ??????????????????????

library(stringdist)  
  
simmilarity\_index <- function(x, y){  
   
 sim\_index <- 1- stringdist(x, y, method = "lv")/max(nchar(as.character(x)), nchar(as.character(y)))  
 return(sim\_index)   
}  
  
for (i in 1:nrow(edge\_list)){  
 for (j in 1:i){  
 if (simmilarity\_index(edge\_list$a[i],edge\_list$a[j]) > 0.8){  
 edge\_list$a[i] <- edge\_list$a[j]  
 }  
 }  
}  
  
for (i in 1:nrow(edge\_list)){  
 for (j in 1:i){  
 if (simmilarity\_index(edge\_list$b[i],edge\_list$b[j]) > 0.8){  
 edge\_list$b[i] <- edge\_list$b[j]  
 }  
 }  
}

Now that the dataset is ready for network analysis, its good to save it as a new CSV file and use it for the future analysis.

write.csv(edge\_list, file = "edge\_list.csv")

## Social Network Analysis

Now the edge list is ready, we can start the social network analysis by turing the edge list to a graph data using **igraph** library. Since it is a collaboration network, it should be a non-directod network.

library(igraph)  
  
graph=graph.edgelist(as.matrix(edge\_list),directed=FALSE)

some basic network analysis measures like density and ????? of graph can be calcuatled right away.

graph.density(graph)

Now a few different centrality measures are calcualted and put next to each other in a table. I have summarised some of the most frequently used centrality measures in the table below with how they are calculated, and their potentials and limits.

|  |  |  |
| --- | --- | --- |
| Measure | Definition | Calculation |
| total degree centrality | Is the number of ties (edges) a node has without considering their directions. It shows the extend a node holds all of the ties in a network. | The degree centrality of a vertex v, for a given graph G := (V,E) with V vertices and E edges, is calculated as It is measured by number of edges of a node. It can be also normalized by dividing by total number of edges. |
| closeness centrality | It is based on the length of the average shortest path between a vertex and all vertices in the graph.Is a measure of the degree to which an individual is near all other individuals in a network. |  |
| betweenness centrality | Betweenness centrality of node v is equal to the number of shortest paths from all vertices to all other that pass v. It is a measure of the extent to which a node is connected to other nodes that are not connected to each other. In the other words the total amount of flow it carries if flow between all other nodes passes it. | The betweenness centrality of a vertex v for a given connected graph G is: Where σst is the total number of shortest paths from node s to node t and σst(v) is the number of those paths that pass through v. It is between 0 and 1 with 1 being the highest centrality. |

They all measure the consequences of having (or not having) a certain node in a graph which shows the importance/power/influence of the node in the network structure.

### High Degree Centrality:

A node with high degree centrality, has more autonomy from its alters (comparing to a node with lower degree), therefore, it has more flexibility/power/opportunities/choices to pick any of those nodes in for example a trade context.

### High Betweenness Centrality:

A node with high betweenness centrality has a large influence on the transfer of items through the network, under the assumption that item transfer follows the shortest paths. Betweenness Centrality is related to connectivity in the graph, in so much as high betweenness vertices have the potential to disconnect graphs if removed. Therefore, a node with high betweenness centrality has the capacity to broker contacts among other nodes, and isolate other nodes or prevent contact.

### High Closeness Centrality:

A node with high acts as a reference point by which can reach a larger number of other nodes at shorter path lengths. It emphasizes the distance of an actor to all others in the network (as oppose to direct ties in degree centrality).

### Potential Limitation:

**Degree centrality** is particularly useful when we want to compare networks cohesiveness as in how the ties have been distributed. It addresses the problem that can rise when graph density is not adequate measure (for example if in a network certain node has disproportionately high degree).

However, It’s main limitation is that it only takes into account the immediate ties that a node has to its egos, rather than indirect ties to all other nodes. A node with a high total degree centrality might be connected to some whole other nodes, but those nodes might be rather disconnected from the network as a whole, thus the ego node is only central locally and therefore doesn’t have the ability to broker between groups (or information/flow originated in other parts of the network is likely not to reach it).

**Betweennesss centrality** is a good measure to find nodes which bridge subgroups in a network, and consequently indicate structural holes in a network. It can also identify levels of hierarchy in a network. If one eliminates all the actors with no betweenness (that is, the “subordinates”), some of the remaining actors will then have 0 betweenness—they are at the second level of the hierarchy.

While it is useful for analyses of spread of disease and pandemics, it can be misleading for simpler analysis.

As **closeness Centrality** relies on the sum of the geodesic distances form each actor to all the others, it can be misleading in complicated graphs. A node that is very close to a relatively small subset of a network, can have the same score as a moderately close node to every actor in a large subset of the network.

I rather pick my favorite measure based on the analysis that I am doing instead of having a universal one. As mentioned above, for picking the most popular node, I will use degree centrality. To see which node is passing the most flow I will use betweenness centrality. For analysis similar to market/resource access, I will use closeness centrality.

It is always informative to sort the table. I picked the in-degree to sort the table.

#calculating different degrees  
degrees <- data.frame(degree = degree(graph),  
in\_degree = degree(graph, mode = c("in"), loops = FALSE, normalized = FALSE),  
out\_degree = degree(graph, mode = c("out"), loops = FALSE, normalized = FALSE),  
btwn= betweenness(graph, directed = T), #Betweenness Centrality  
close = closeness(graph, mode = c("all")), #Closeness Centrality  
eigen <- evcent(graph), #Eigenvector Centrality   
bon <- bonpow(graph) #Bonacich Power  
)  
  
#cleaning up the table  
degrees = degrees[,c(1:6, 28)]  
degrees\_sorted <- degrees[order(-degrees$in\_degree),]

If you want to export the dataframe to a nice pdf file, you can use the **"gridExtra"** library.

library(gridExtra)  
pdf("data\_output.pdf", height=15, width=15)  
grid.table(degrees)  
dev.off()

cor(degrees\_sorted)  
   
# regress attributes on centrality measures  
in\_degree = degree(graph, mode = c("in"), loops = FALSE, normalized = FALSE)  
out\_degree = degree(graph, mode = c("out"), loops = FALSE, normalized = FALSE)  
total\_degree = degree(graph)  
btwn= betweenness(graph, directed = T)  
summary(lm(total\_degree ~ btwn))

## Social Network Graphs

## start the graph ##  
set.seed(12)  
l <- layout.kamada.kawai(graph)  
  
# Size of node by in-degree.  
V(graph)$size <- 15\*(degree(graph, mode="in")/ max(degree(graph, mode="in")))  
  
# Size of node label by in-degree.  
V(graph)$label.cex <- ((betweenness(graph, directed = T)+1)/(max(betweenness(graph, directed = T))+1))\*2  
  
# plot the graph  
plot(graph, layout=l, edge.arrow.size=.2, edge.curved=T, edge.color="grey")  
title("Co-Authorship Graph",cex.main=0.8,col.main="black")

### Community extraction & Clustering

fgn = edge.betweenness.community (graph, directed = TRUE, edge.betweenness = TRUE, merges = TRUE,  
 bridges = TRUE, modularity = TRUE, membership = TRUE) ## run Girvan-Newman partitioning  
  
plot(fgn, graph, layout=l, edge.arrow.size=.2, edge.curved=T, edge.color="grey") ## plot G-N partitioning  
  
fwt <- walktrap.community(graph, steps=200,modularity=TRUE) # , labels=TRUE) ## run random walk partitioning  
  
plot(fwt, graph, layout=l, edge.arrow.size=.2, edge.curved=T, edge.color="grey") ## plot R-W partitioning  
  
flp = label.propagation.community(graph) ## run label propogation partitioning  
  
plot(flp, graph, layout=l, edge.arrow.size=.2, edge.curved=T, edge.color="grey") ## plot L-P partitioning  
  
#comparing different community dittection algorithms  
compare(fgn, fwt, method= c("nmi"))  
compare(fgn, fwt, method= c("rand"))  
compare(fgn, fwt, method= c("adjusted.rand"))  
  
compare(fgn, flp, method= c("nmi"))  
compare(fgn, fwt, method= c("rand"))  
compare(fgn, flp, method= c("adjusted.rand"))  
  
## get the results in a dataframe  
  
girvan = data.frame(fgn$membership)  
rw = data.frame(fwt$membership)  
flpm = data.frame(flp$membership)  
  
degrees <- cbind(degrees, girvan, rw, flpm)