Organizational Network Analysis Using R-igraph

Christine Sowa, Tony Johnson, and Ian McCulloh

November 28, 2016

Executive Summary

This tutorial reviews some basic concepts in Organizational Network Analysis (ONA). It provides sample code in R to calculate some general measures and produce basic visualization. The necessary packages are invoked with the commands:

```
install.packages("igraph")
install.packages("data.table")
library(igraph)
library(data.table)
```

ONA can be defined as the study of complex human systems through the mapping and analysis of social relationships between people, groups, and/or organizations. It can be used by the operations research analyst to reveal the structure, composition, and characteristics of existing networks. With this approach, the savvy analyst is prepared to study and refine the communications and information flow of friendly networks. This builds resiliency while disrupting and fragmenting opposing networks in order to build disunity and ultimately collapse. (Bell and Conjar 2016)

Organizational Network Analysis (ONA)-Examples

The purpose of this document is to help operations research analysts understand how to use Organizational Network Analysis (ONA), and what can be achieved by using R as a tool for the analysis. It provides the background and an initial approach to using ONA for assessing and making recommendations to improve the effectiveness and performance of an organization. It is not meant to be a detailed technical guide; however, it is assumed that the reader has some familiarity with community development techniques and networks. Software code used to conduct basic analysis is provided using R.

The institution from which the emails (over 97,000) were collected consists of a major activity division/directorate (MADD) of 250 employees divided into six different departments. Each department has between 50 and 65 employees. Thus, a sub grouping can be drawn from the MADD based on departmental affiliation.

Data Preparation

Email data can be gathered from a Microsoft exchange server and is most useful if it contains columns that are arranged in the following formats:

```
pid1        pid2      num_emails
105          1010          3
105          1044          17
1037         101           49
```

and

```
pid     dept_group
1010    dept-001
1011    dept-002
1047    dept-005
```

The first file contains the edge information or links necessary to show who sent emails to whom while the second file contains the vertices or nodes. The nodes file can be used to provide an annonymized table of attributes that can be used to describe various properties of each node such as which department they belong to, or their academic background.

Data Import

While R can read excel .xls and .xlsx files sometimes these file types cause problems because of hidden codes embedded in them. Comma separated files (.csv) are much easier to import. It is best to save these files as .csv files before reading them into R. The best way to read in a csv with R is by reading the file into the R environment as a `data.table`. Data table is an extremely useful R package that uses a modified R data frame to import csv data and other file formats at remarkable speed. To learn more about data tables and their usefulness in importing large amounts of data go to https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.pdf. We can read our comma separated vertices and edges into two separate R data tables. Here in the example we will use the fread command using the "sep" and "header" options. There are many other options that can be used as well.

```
## Load data frames data.a and data.b
data.a <- fread("../ONA/ONA-Data/nodes.csv",sep=",", header=T)
data.b <- fread("ONA-Data/emailedges.csv",sep=",", header=T)
```

The Network

Look at the entire network of email traffic between the 250 employees of the university below. It is difficult to derive any useful information about the network or its email traffic through visual inspection. R can be useful at generating Figure 1, but further analysis is needed to gain insight into the organization and its characteristics.

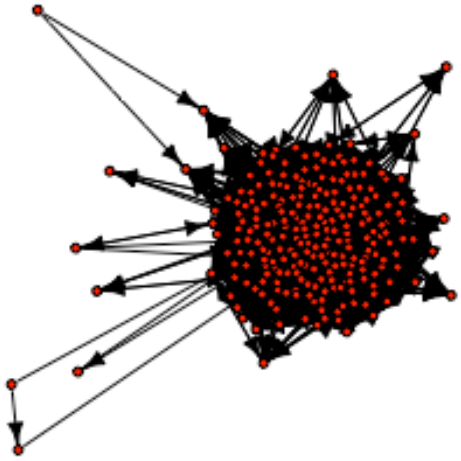


Figure 1: Email connections between all members of the network

ONA Techniques, Tactics and Procedures

The first step in the ONA process for this example is to extract the network of interest, i.e. the employees within the department, out of the larger set of data consisting of employees from the entire sector. The full data set includes individuals and departmental groups that reside on the same network but are outside the boundary conditions for the ONA of interest. Here we can use the properties of data tables to filter down to supgroups of interest:

```
## Parse out subgroups of interest for analysis by using filtering on
the code that corresponds to the departments we are interested in
analyzing.
subgroup1<-data.a[dept_group=="AAA-001"]
subgroup2<-data.a[dept_group=="AAA-002"]
```

A custom function can be written to extract a sub grouping of the network from the larger network to create a dichotomous group network at a certain threshold. We call our custom function extractSubnet.

```
extractSubnet<-function(subgroup, edges, cutoff_value){
  ## This function takes a subgroup of nodes from a larger network
```

```
  ## and create a dichotomous network consisting only of nodes from
  ## within the group at a certain edge threshold.
  ## INPUT:
  ##     subgroup : list of nodes within the subgroup
  ##        edges : data.table consisting of all edges in the network
  ##                containing at least three columns
  ##                source, target, and edge weight
  ## cutoff_value : scalar value for threshold cutoff point
  ##
  ## OUTPUT:
  ##            g : network object containing all nodes of subgroup
  ##                and edges only where both the source and the
  ##                target are from the subgroup
  ##
  dt.edges <- edges # Assign all edges to a data.table

  # Use %in% to search and select only nodes within the subgroup
  dt.edges <- dt.edges[pid1 %in% subgroup$pid]
  dt.edges <- dt.edges[pid2 %in% subgroup$pid]

  # Use same convention to select only those rows which meet the
cutoff_value
  dt.edges <- dt.edges[num_emails >= cutoff_value]

  links <- as.matrix(dt.edges)

  # Build the network object g
  g <- graph_from_data_frame(d=links, vertices=subgroup, directed=T)

  return(g)
  ## end function extract_subnet
}
```

The function **extractSubnet** can be used to extract a group network from the larger network provided by the exchange administrator by passing the function the group name, the larger data file, and a cutoff threshold. The cutoff threshold is a value that specifies the number of emails that must be sent between individuals to qualify as a relationship in the network. Note that the number of emails between individuals is more indicative of their personal email behavior than it is the strength of relationship. The goal of using emails for network data is to identify the likely presence of regular communication. Setting a threshold to reduce data complexity to a level where network structure can be observed provides an estimate of organizational collaboration.

The function **extractSubnet** can now be used to identify several views of the network at different complexity levels or thresholds. Below are three such views. The naming convention is designed to include the cut off value.

```
subnet.5 <- extractSubnet(subgroup2, data.b, 5)
subnet.11 <- extractSubnet(subgroup2, data.b, 11)
subnet.18 <- extractSubnet(subgroup2, data.b, 18)
```

Basic Visual Analysis

Now that we have a way to extract a sub group of nodes along with the corresponding edges between them, we can visualize the sub network in R by using the plot commands. Note that one of the final tasks accomplished by our function **extractSubnet** was to create a network object (g) which defines the resulting network in terms that the plot command understands. This preconditioning makes visualizing the network a simple one line command. The plots shown in Figures 2-4 are for different views. Note the options to adjust the size of the vertices and arrows as well as to change the color of the vertices. Other visualization options will be discussed later.

```
plot(subnet.5, vertex.color="red", vertex.size=4, vertex.label=NA,
edge.arrow.size=0.3, edge.color="black")
```

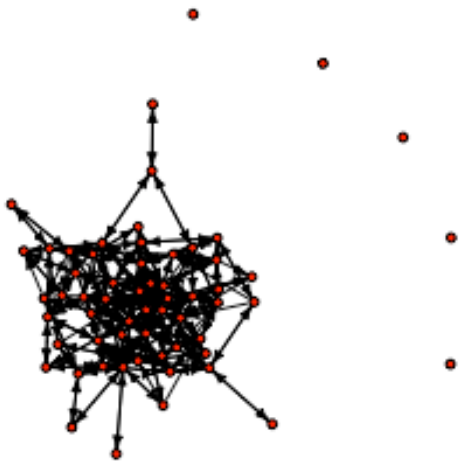


Figure 2: Plot showing network where each edge is a connection between two nodes that exchanged at least 5 emails.

```
plot(subnet.11, vertex.color="red", vertex.size=4, vertex.label=NA,
edge.arrow.size=0.3, edge.color="black")
```

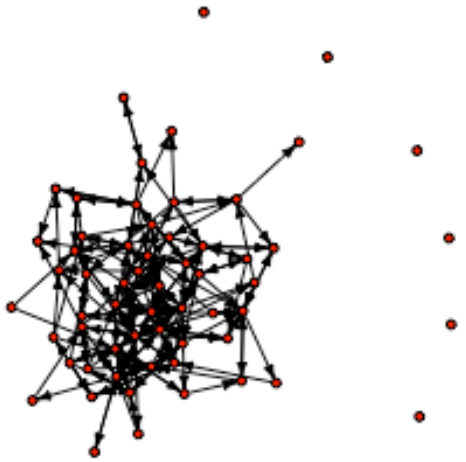


Figure 3: Plot showing network where each edge is a connection between two nodes that exchanged at least 11 emails.

```
plot(subnet.18, vertex.color="red", vertex.size=4, vertex.label=NA,
edge.arrow.size=0.3, edge.color="black")
```

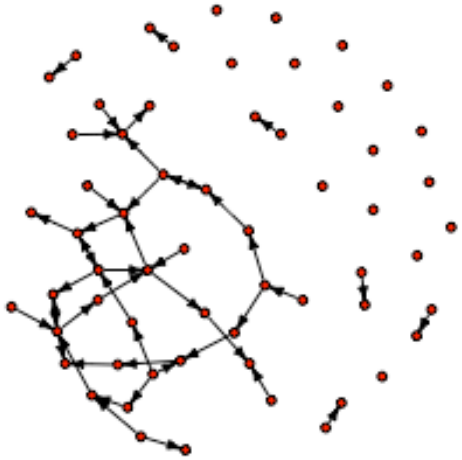


Figure 4: Plot showing network where each edge is a connection between two nodes that exchanged at least 18 emails.

Analysis

The visualization of the nodes on the network immediately provides some interesting insights without any additional computations. If we choose to look at, say, subnet.18, we can get an overview of the attributes using the summary command, a summary of the node attributes with the V()[[]] command, and a look at the edge labels with the E()[[]] command.

```
summary(subnet.18)

## IGRAPH DN-- 60 46 --
## + attr: name (v/c), dept_group (v/c), num_emails (e/n)

V(subnet.18)[[1:5]]

## + 5/60 vertices, named:
##   name dept_group
## 1    2    AAA-002
## 2    6    AAA-002
## 3    8    AAA-002
```

```
## 4   17    AAA-002
## 5   24    AAA-002
```

```
E(subnet.18)[[1:5]]
```

```
## + 5/46 edges (vertex names):
##   tail head tid hid num_emails
## 1  100  224  22  57         20
## 2  159  224  36  57         18
## 3  171  224  40  57         20
## 4  224  100  57  22         19
## 5  110   42  26   8         19
```

We can determine the number of isolate nodes in the subnet.18 graph by counting the number of nodes that have a degree of 0.

```
sum(degree(subnet.18)==0)
```

```
## [1] 15
```

Here, a quarter of the nodes (15/60 = 25%) are isolates, meaning that they have no interaction (above the 18 count threshold we had set previously) over email with any of the other nodes in the group. We can deduce from the data that these nodes may work offsite in direct support of clients and may not interact with others in their own group with a high frequency. If they do interact with each other, it may be done offsite and through non-organizational email addresses. Further investigation reveals that many people in the connected cluster have no idea who the pendant nodes are or that they work for the same employer. This may be a problem for an organization that wishes to be pioneering and sharing knowledge and resources. Lessons learned in support of one client have limited ability to transfer and benefit other clients. Data complexity (the network is too dense) is not an issue in this organization. This is one example of how ONA with R can be used to quickly analyse organizational data to provide insight into the underlying network. Further analysis will be focused on the full, non-reduced network.

Organizational Structure Analysis and Graph Level Diagnostics

Organizations exist to achieve goals. These goals are broken down into tasks as the basis for jobs. Jobs are grouped into departments. Departments in organizations may be characterized by marketing, sales, advertising, manufacturing, and so on. Within each department, even more distinctions can be found among the jobs people perform. Departments are linked to form the organizational structure. The organization's structure gives it the form to fulfill its function in the environment (Nelson and Quick 2011). The next example and the set of graph diagnostics that follow apply to the organization as it relates to the structure of its group dynamics.

Reciprocity refers to responding to a positive action with another positive action; it creates, maintains and strengthens various social bounds. It is the foundation of social order and is a major key to success. This applies not only in social networking

but also in all rounds of human activities. The potential for reciprocal actions by players increases the rate of contribution to the public good; reciprocity is a form of social obligation and is a motivation for returning favors from others (Fehr and Simon 2000).

The reciprocity metric is found using the following command.

```
rec<-reciprocity(subnet.18) #The proportion of reciprocated ties (for a directed network).
print(rec)

## [1] 0.1304348
```

The next set of metrics focus on network (graph) level properties. Using the transitivity function to find the clustering coefficient, the diameter function, and density of the network can be calculated using the commands:

```
transitivity(subnet.18)

## [1] 0

diameter(subnet.18) # returns the distance of the longest geodesic distance (length of the shortest path between two nodes)

## [1] 10

edge_density(subnet.18) #The proportion of present edges from all possible edges in the network.

## [1] 0.01299435
```

The clustering coefficient, found using the transitivity function (http://igraph.org/r/doc/transitivity.html) is similar to density; however, it indicates the tendency for nodes to cluster around each other. In other words, the nodes that are not isolates do not form transitive ties as they cluster together. Too high of a clustering coefficient (>0.75) indicates a trend towards a fully connected clique. This leads toward group think and lacks the diversity for an agile organization. A zero clustering coefficient means that there are no transitive ties in the network; therefore, there are no cohesive triads. High clustering coefficient is a poor quality of an efficient organization, because there is a high number of lateral peer-to-peer ties which take time and resources away from purely efficient organizational structures. Low clustering coefficient is good for lean organizations, but poor for agile ones. A moderate clustering coefficient is advisable for agile organizations in a pioneering market stage. The scale of "moderate" is dependent upon the context and size of the network.

A low diameter (<4) indicates that information can easily and quickly move from one side of the network to the other. The diameter measure does not scale well with network size. Given that an individual's awareness of resources and knowledge across the network decays somewhere between 2-3 steps through the network, a

diameter greater than or equal to 6 will indicate limits to knowledge and resource exchange. Such a high diameter in this case indicates that information may take a long time to pass from one side of the network to the other.

The density is very low, which is the result of the high number of isolates. Density also scales poorly with network size. It also has the mathematical equivalence to average degree. Average degree is equal to the density times the network size. Networks with high average degree can become problematic in that nodes are too busy maintaining network ties to have meaningful interactions or generate useful ideas. Networks with low average degree indicate missed opportunities for knowledge and resource exchange. The range of "good" average degree is dependent upon the context of the network relation and the nature of interaction.

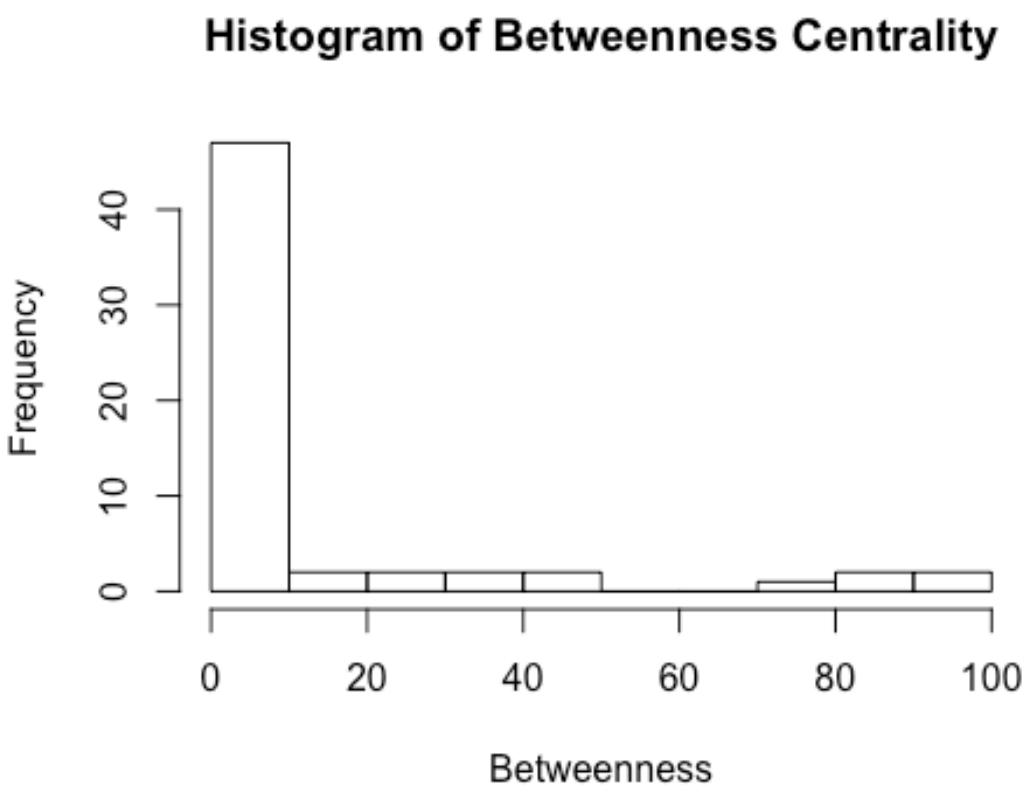
Agent Level Centrality Diagnostics

Centrality measures are important measures of influence and opinion leadership within the organization. The two most important centrality measures are betweenness centrality and eigenvector centrality, and they can be calculated using the following commands:

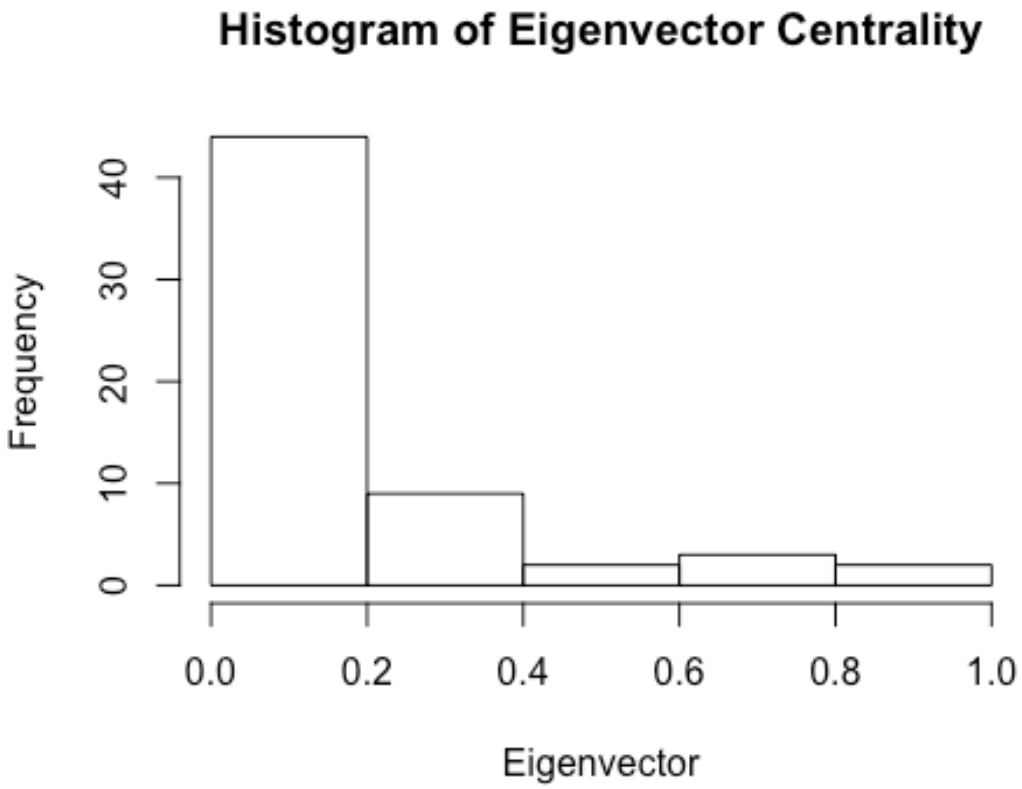
```
b<-betweenness(subnet.18)
e<-eigen_centrality(subnet.18)
```

These values are calculated for each node in the network. Therefore, it is impractical to list individual values. The distribution of these values can be represented with a histogram, however.

```
hist(b,main = paste("Histogram of Betweenness Centrality"),xlab =
"Betweenness")
```




```
hist(e$vector,main = paste("Histogram of Eigenvector Centrality"),xlab
= "Eigenvector")
```



The eigenvector centrality values for this network are skewed to the left, with most of the eigenvector centrality values being very low. This indicates that we can isolate the nodes with high eigenvector centrality to see how they can spread through the network. An unfortunate property of eigenvector centrality, however, is that it tends to only identify brokers in a single network cluster. When there exist multiple clusters, central power brokers in other clusters are not properly valued.

In order to analyze the network without considering isolates, it is necessary to remove them. This is conducted in two steps. The first step is to identify the isolates, then remove them. This is completed using the following commands:

```
delete.isolates <- function(graph, mode = 'all') {
  isolates <- which(degree(graph, mode = mode) == 0)
  delete.vertices(graph, isolates)
}

small.group<-delete.isolates(subnet.18, mode = 'all')
```

The resulting network can be observed

```
plot(small.group, vertex.color="red", vertex.size=4, vertex.label=NA,
edge.arrow.size=0.5, edge.color="black")
```

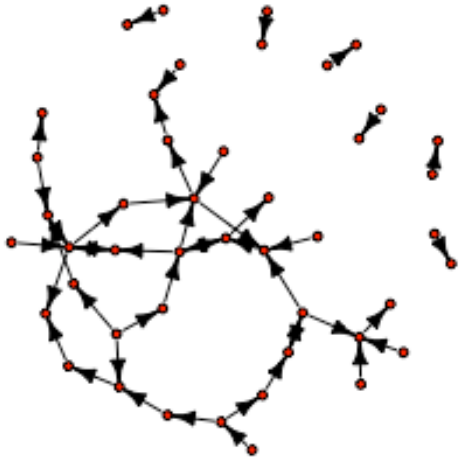


Figure 5: The network with isolates removed.

Degree centrality is not particularly interesting on its own. The number of connections that a node has does not really identify its importance apart from the fact that degree centrality is often mathematically correlated with other centrality measures that are often more important. When the measures are correlated, they often conform to intuition. It is when measures are uncorrelated that ONA can provide interesting, counter-intuitive findings. For example, a node that is high in betweenness centrality, yet low in degree, is a power broker or gate keeper. This example does not provide an example of this type of power broker; however an example dataset of how to plot this type of analysis is provided in Figure 6.

```
plot(x=degree(subnet.18),
     y=betweenness(subnet.18),
     pch=19,
     cex=1,
     xlab="Degree",
     ylab="Betweenness",
     col="red",
     family="Arial Black" )
```

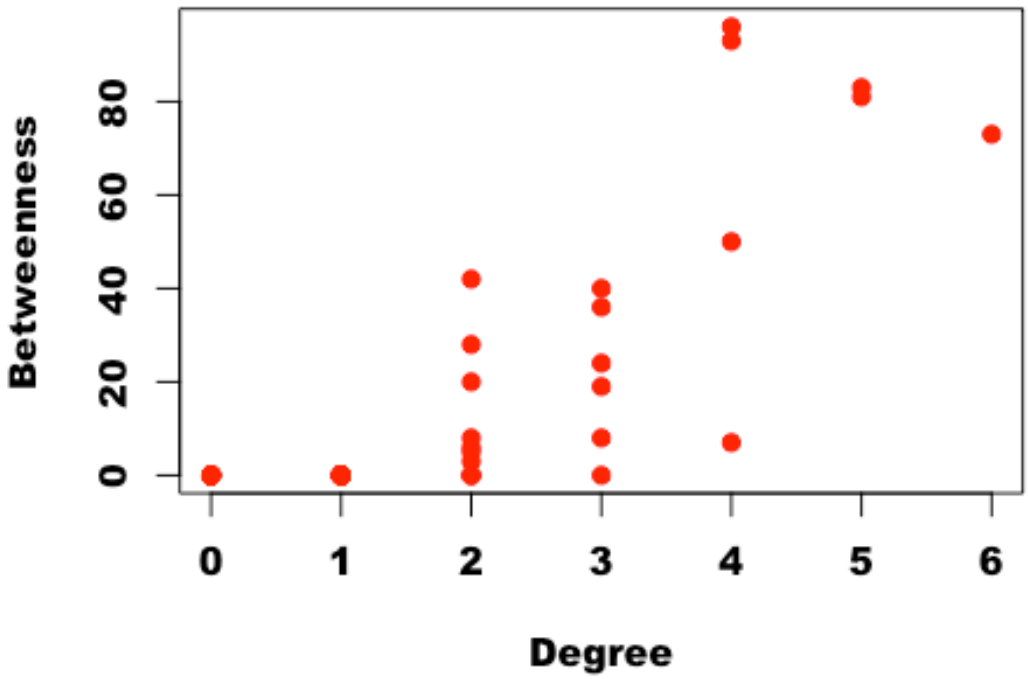


Figure 6: Graph showing the network's betweenness compared to its degree.

Nodes occupying positions in the upper left portion of Figure 6 are of more interest from an informal power broker perspective. For denser networks, eigenvector centrality should be substituted for betweenness centrality. This network shows little by way of informal power brokers.

Visualization

Visualization options are important tools not only for analysis, but for communicating results in sufficient bandwidth to convey rapid understanding to those who must make decisions based on the information. There are seven dimensions of visualization that can be used to convey information: horizontal position, vertical position, size, color, shape, intensity, and texture. Some dimensions are better than others for communicating different types of information. For example, size is more useful for communicating numeric data, while shape is often better for categorical data. It is not cognitively possible to communicate more than seven dimensions of data without resorting to multiple visualizations or displays of the data. For network analysis, we recommend the following conventions:

Horizontal and vertical position

Horizontal and vertical position are used to maximize the layout of the network. There are multiple layout algorithms in R. A detailed overview of each is beyond the scope of this paper. However, more information about R network layout algorithms can be found at http://kateto.net/network-visualization.

Size

Size is used to represent a numeric node value, such as betweenness centrality, seniority, education level, or some other hypothesized measure of importance. When using centrality as the basis for size, central nodes can sometimes be displayed so large, that the network structure cannot be adequately observed. In this case it is recommended to force a logarithmic scaling to the centrality measures to convey proper understanding.

Color

Color is used to represent group/cluster affiliation, or some attitudinal variable such as workplace satisfaction. Colors are defined in R as follows,

```
#1=black
#2=red
#3=green
#4=blue
#5=light blue/aqua
#6=pink
#7=yellow
#8=gray
```

Shape

Shape is used to represent a categorical node value, such as gender, job role, location.

Intensity

Intensity is used to focus attention at key network terrain and/or for over time visualization.

Texture

We should refrain from using texture unless another dimension of data is absolutely necessary. The natural texture of the network itself can often obscure the meaning communicated by other visualization variables.

Figure 7 provides an example visualization of the **subnet.18** network now labeled **small.group** as above because the isolate nodes are removed. The label font size is defined and the color is specified as blue. The color of the nodes is red and the links are black. Here, the nodes are sized by betweenness, and scaled to provide a better

visual effect. The layout uses a spring-embedded or force-directed layout algorithm. The supporting command line code is as follows, and can be written in a script. Note that the vertex and edge attributes can be explicitedly stated as one pre line versus putting them all in one long command. The **small.group** variable is actually a network object that can be changed and manipulated by various commands. Again, the plot command is used to graph the object.

```
bet.18 <- betweenness(small.group)
# Vertex attributes
V(small.group)$size <- 4 + bet.18/6
V(small.group)$color <- "red"
V(small.group)$label.color <- "blue"
V(small.group)$label.cex <- .7
V(small.group)$label.dist <- 1
# Edge attributes
E(small.group)$color <- "black"
E(small.group)$arrow.size <- 0.4

plot(small.group,main="Size Vertices by Betweenness")
```

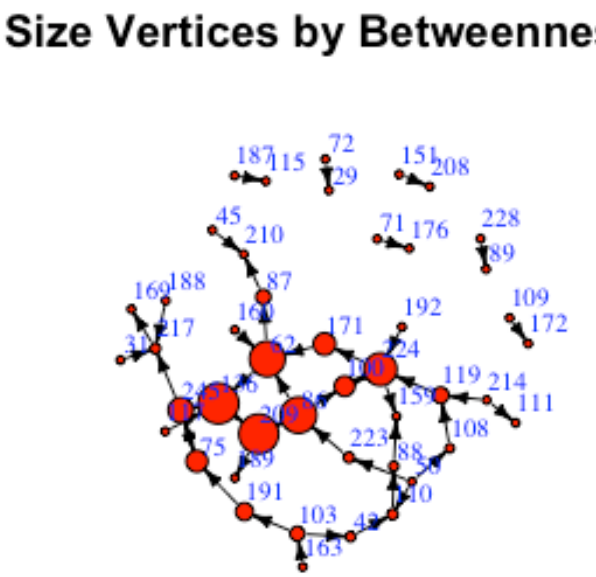


Figure 7: Network plotted with the nodes sized by betweenness.

Another visualization is to treat the network as if the network were undirected. For this interpretation, the assumption is that if there exists an email from actor *i* to actor *j*, then a relationship is assumed to exist between both *i* and *j*, as well as *j* and *i*. In other words, there is simply a relationship between them and direction does not matter. Figure 8 is an example of this visualization with supporting modification to the code.

```
plot(small.group, edge.arrow.size=0, vertex.label=NA)
```

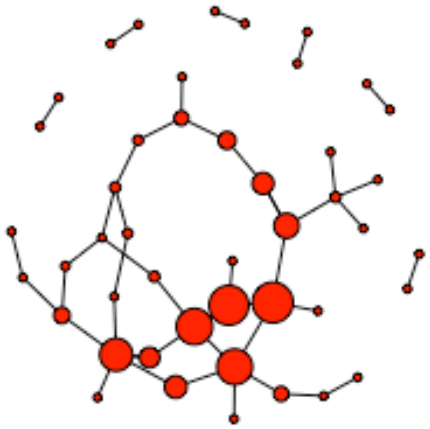


Figure 8: Network sized by betweenness and plotted without node labels.

In Figure 8, the labels are omitted, arrows are hidden, and the node and edge colors are modified. These visualizations may lack impact to the reader at this point. This is due to the fact that the reader will have no context by which to understand the organization and the interpersonal dynamics within it. Labels can sometimes introduce bias and detract from the true structure of the underlying organization. When node labels identify key power brokers in the organization, they often confirm intuition and drive inquiry into areas of the organization that are more effective or problematic. This information will cause changes in behavior among those actors that are exposed to the labeled network diagrams. Some change may be positive, but others may not. Some actors may experience diminished performance, feeling on the periphery of the network, even if they provide an important function

for the organization. For these reasons, it is important to be careful about displaying node labels. Visualization options that hide node labels are important for facilitating discussions about the organization without adversely affecting node behavior.

The network in Figure 9 is an alternate approach to visualization, where the labels are sized based on their centrality value and the edges are gray to provide contrast. Code for this implementation is as follows.

```
bet.18 <- betweenness(small.group)
# Vertex attributes
V(small.group)$size <- 4
V(small.group)$color <- "red"
V(small.group)$label.color <- "black"
V(small.group)$label.cex <- .7 + bet.18/60
V(small.group)$label.dist <- 1
# Edge attributes
E(small.group)$color <- "light gray"
E(small.group)$arrow.size <- 0.4

plot(small.group,main="Size Vertices Labels by Betweenness")
```

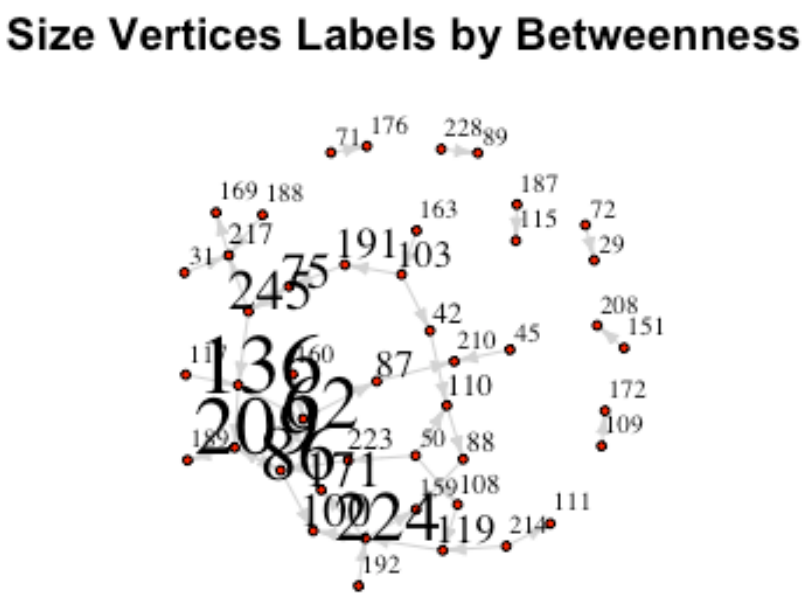


Figure 9: Network displayed with node labels sized by betweenness.

Cohesive Groups and Cluster Analysis

So far, this ONA process has focused on individuals within a single group. In many organizational settings, especially matrix organizations, individuals must work across groups. When one manager is responsible for bringing diverse skills and talent together to complete a project and other managers are responsible for developing and maintaining organizational capability, the employees find themselves working for more than one boss. Each boss will likely have different goals and priorities. As employees collaborate, attempt to provide value, and develop a shared understanding of value, it may not be communicated back up the chain of leadership. It may not even be clear where these pockets of communication even exist within the organization.

Managers may have a more difficult time understanding value in these more complex organizations. Who holds opinion leadership within these informal networks that may shape and define employee perceptions of value? Are these views in line with management goals and priorities? Which sub-organizations interact regularly? Are there certain sub-organizations that are more central or influential than others? These questions are addressed with group level ONA.

Each node in the network must be assigned an attribute to identify which group they belong to. In this example data, we will focus on one sector within the organization that contains multiple groups. This is of value for expository purposes, because it requires extracting the sector from the larger data set, while still assigning a group identification variable for later analysis. The sample code is provided. The nodes that are part of the sector must be identified and be associated with a variable indicating their group affiliation. The code in the following section extracts the sector level node-attribute list.

Truss Clustering Method

One method to find clusters in a network is through subgraphs called trusses (Cohen 2009). A k-truss is defined as, "a non-trivial, one-component subgraph such that each edge is reinforced by at least k–2 pairs of edges, within the subgraph, making a triangle with the that edge." Additionally, "a maximal k-truss is a k-truss that is not a proper subgraph of another k-truss." The larger the value of k, the more inter-connected a subgraph is within itself which makes it interesting to use the maximal k-truss to find very cohesive groups. The following code is a function that takes an iGraph object, g, and an integer, k, as an input and returns a subgraph for the network. It does this iteratively.

```
calcTrusses <- function(g,k){
  ##
  ## This function takes an R iGraph object, g, and returns a
  ## hierarchical (non-empty) subgraph that represent the cores of a
  ## network at different levels of granularity, k.  The cohesive
  ## subgraphs are relatively small sub-structures and may be
  ## scattered all over the graph
```

```r
  ##
  ## INPUT:
  ##            g : an undirected igraph representing the network
  ##            k : scalar representing the desired level of
  ##                granularity or k-truss level.  Given a
  ##                graph, G, the k-truss of G is the largest
  ##                subgraph of G in which every edge is
  ##                contained in at least (k-2) triangles
  ##                within the subgraph [J. Cohen 2008].
  ##                k must be at least 3.
  ##
  ## OUTPUT:
  ##            g : network object containing the k-truss
  ##                cohesive subgraph of G or an empty graph
  ##                which can be used to signal no truss at
  ##                that k level.  Test comment
  ##

  # Validate input igraph object: must be undirected.
  if (igraph::is.directed(g)){
    stop("Graph must be undirected.")
  }

  edge_count <- igraph::ecount(g)

  while (edge_count != 0){
    A <- igraph::get.adjacency(g)
    # The edge i-j occurs in number of triangles equal to scalar
product of
    # row i and row j in the adjacency matrix A.
    triangle_matrix <- Matrix::tcrossprod(A)  # sparse matrix
    # Each edge needs to be in at least (k-2) triangles.
    support_matrix <- as.matrix(triangle_matrix >= (k - 2))
    # Look up support for each edge.
    has_support <- support_matrix[igraph::ends(g, igraph::E(g))]
    # Trim to subgraph of supported edges.
    g <- igraph::subgraph.edges(g, which(has_support))
    # Re-compute number of edges in subgraph and see if it was trimmed.
    edge_count_new <- igraph::ecount(g)
    if (edge_count == edge_count_new) {
      break
    }
    edge_count <- edge_count_new
  }
  return(g)
}
```

Given the function to calculate the trusses in a network, the next step is to apply it to a data set. For this example we will continue using the subnet.18 dataset from above.

```
#g <- as.undirected(subnet.5, mode = "each")
#g <- simplify(subnet.5, remove.multiple = T, remove.loops = T)

g <- delete.isolates(subnet.5, mode = 'all')
g <- as.undirected(g, mode = "collapse")

#base.color <- "gray95"
base.color <- NA

E(g)$eid <- seq(ecount(g))
V(g)$vid <- seq(vcount(g))

lout <- layout_with_lgl(g)

plot(g, vertex.size=4, vertex.label=NA, edge.arrow.size=0.3,
edge.color="gray")
title(main=paste("subnet.5 has",ecount(g),"edges."))
```

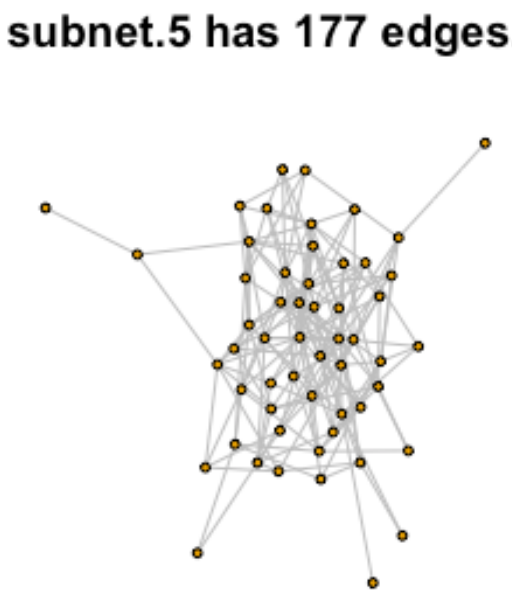
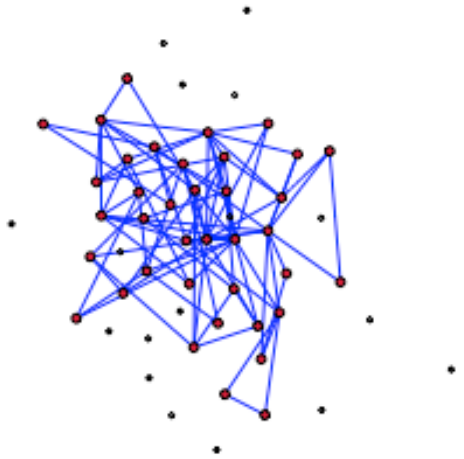
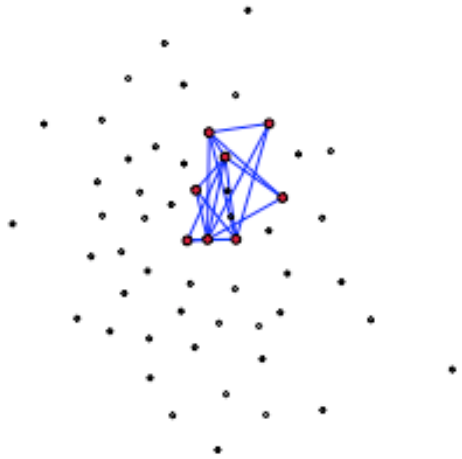


Figure 10: Plotting the subnets of the network as the algorithm runs through it.

```
k <- 3
repeat{
  g.truss <- calcTrusses(g,k)
  num.tri <- length(count_triangles(g.truss))
  if (vcount(g.truss)==0){break}

  g.truss.eid <- E(g.truss)$eid
  g.truss.vid <- V(g.truss)$vid

  E(g)$color <- base.color
  V(g)$color <- base.color
  E(g)$weight <- 1
  V(g)$weight <- 2

  E(g)[g.truss.eid]$color <- "blue" # blue
  V(g)[g.truss.vid]$color <- "#DC143C" # Crimson
  E(g)[g.truss.eid]$weight <- 1
  V(g)[g.truss.vid]$weight <- 4

  plot(g,
       vertex.size=V(g)$weight,
       edge.width=E(g)$weight,
       vertex.label=NA, layout=lout)
  title(main=paste(k,"- truss\n",ecount(g.truss),"edges
and",num.tri,"triangles"))
  k <- k + 1
}
```

3 - truss
90 edges and 38 triangles





Once the base graph is created, the calc.trusses() function finds the relevant k-trusses in the network. The code above starts by setting the initial k value equal to 3. It then goes through an iterative repeat loop that runs the calc.trusses() function, verifies to see that the trusses found are not empty and proceeds to change the color of the graph's truss edges to better indicate their presence in the plot. At the end, it increases the k value by one and runs the code all over again until the maximal trusses are found in the data set.

Conclusion

In this tutorial, we have reviewed some basic concepts in Organizational Network Analysis (ONA). Sample code in R was provided to get you started on calculating some general measures and produce basic visualization. This tutorial is by no means comprehensive as there is a plethora of measures and analysis that can be done on an organization. With some basic knowledge and practice, much can be derived about an organization using tools freely available in the open source software R.

References

Bell, Christopher P., and Elizabeth Conjar. 2016. https://www.boozallen.com/content/dam/boozallen/documents/01.005.14_NetworkAnalysis_WP-webonly.pdf.

Cohen, Jonathan. 2009. "Graph Twiddling in a Mapreduce World." *Computing in Science and Education* 11 (4): 29–41.

Fehr, E., and G. Simon. 2000. "Fairness and Relation; the Economics of Reciprocity." *Journal of Economic Perspectives* 14 (3): 159–81.

Nelson, D. B., and J. C. Quick. 2011. *Understanding Organizational Behavior*. Mason, OH: South-Western Cengage Learning.