Assembling my Network
The ecology is theoretical but the fun is real.

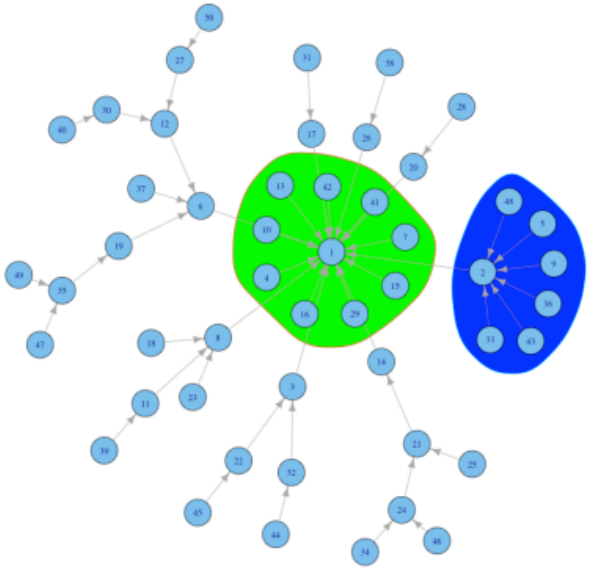
Network basics with R and igraph (part I of III)

Posted on May 29, 2013

In this post I am going to go over some of the basics that I have learned about networks in R by using the igraph package. I originally intended to have this as a reference for myself, but decided to turn it into more of a tutorial for others who, like me, are just getting started with programming in R and interested in networks and network theory.

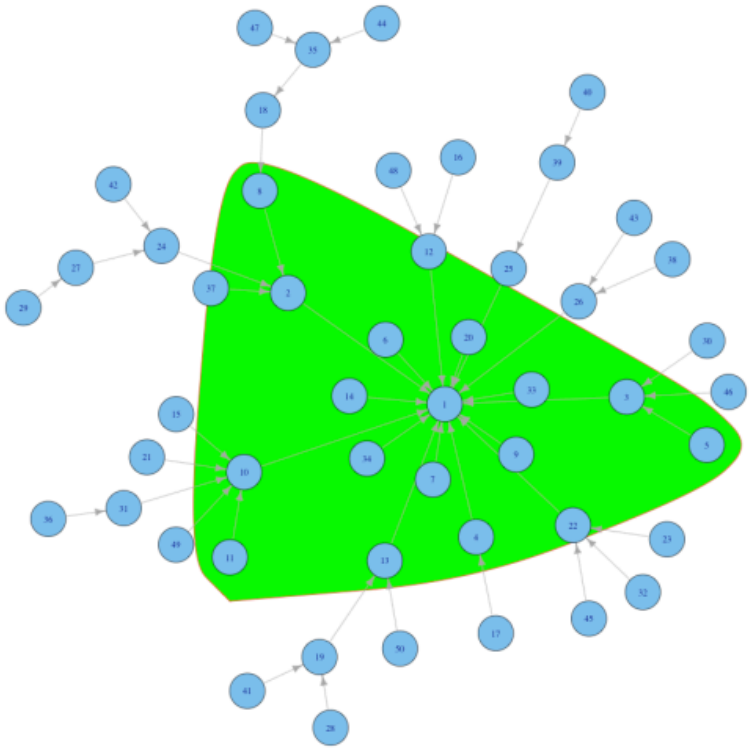
```
1  ################################################################
2  ################################################################
3  ####
4  ####            PART I: NETWORKS
5  ####
6  ################################################################
7  ################################################################
8  # the igraph library is good for network based analyses
9  library(igraph)
10
11 # First lets build a network using basic formulas:
12
13 graph.onelink<-graph.formula(A-+B)
14
15 # This gives us a two species network (A and B) with one link (represented by A-+B).
16 # We can visualize our simple 2 species network with plot.igraph().
17
18 plot.igraph(graph.onelink)
19
20 # Using graph.formula() we can create any graph we want, as long as we are willing t
21
22 graph.foodchain<-graph.formula(A-+C,B-+C,C-+D)
23
24 # and plot it:
25
26 plot.igraph(graph.foodchain)
27 #A and B are eaten by C while D eats C
28
29 #igraph has a function for generating random networks of varying size and connectanc
30
31 graph.random.gnp<-erdos.renyi.game(n=20,p.or.m=.5,type="gnp",directed=T)
32 plot.igraph(graph.random.gnp)
33
34 # We can also change the layout of the graph, here we will plot the nodes in a circl
35 plot.igraph(graph.random.gnp,layout=layout.circle)
36
37 # Here we have created a random directed graph with 20 species ("n") and a connectar
38 # By setting "type='gnp'" we tell the function to assign links with the probability
39 # Similarly we can set the number of links that we want in the system to a value "m'
40
```

```
graph.random.gnm<-erdos.renyi.game(n=20,p.or.m=100,type="gnm",directed=T)
plot.igraph(graph.random.gnm)

# Here the number of links in the network is set to 100, and they are assigned unif

# Rather than being truly random, many real networks exhibit some type of organizat
# To model scale free networks Barabasi and Albert developed the preferential attac

# In igraph we can use the barabasi.game() function:
graph.barabasi.1<-barabasi.game(n=50,power=1)

# For this graph I will introduce some new plotting tools to specify layout and ver

plot.igraph(graph.barabasi.1,
layout=layout.fruchterman.reingold,
vertex.size=10,          # sets size of the vertex, default is 15
vertex.label.cex=.5,     # size of the vertex label
edge.arrow.size=.5         # sets size of the arrow at the end of the edge
)

# there are a number of different plotting parameters see
#?igraph.plotting
#for details

plot.igraph(graph.barabasi.1,
layout=layout.fruchterman.reingold,
vertex.size=10,
vertex.label.cex=.5,
edge.arrow.size=.5,
mark.groups=list(c(1,7,4,13,10,16,15,41,42,29),
c(2,48,5,36,43,33,9)), # draws polygon around nodes
mark.col=c("green","blue")
)
```



Nodes 1 and 2 are highlighted as hubs in this network

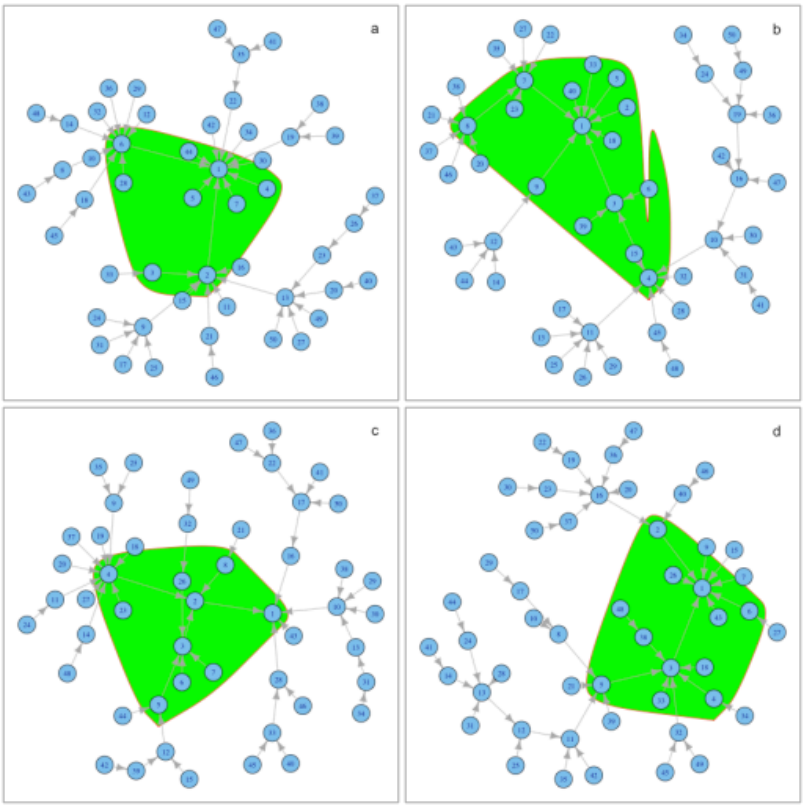
```
# In the above plot a green and blue polygon are used to highlight nodes 1 and 2 as

# We can use a community detection algorithm to determine the most densely connecte

barabasi.community<-walktrap.community(graph.barabasi.1)

# This algorithm uses random walks to find the most densely connected subgraphs.

members<-membership(barabasi.community)
# The members() function picks out the membership vector (list of nodes in the most

par(mar=c(.1,.1,.1,.1))    # sets the edges of the plotting area
plot.igraph(graph.barabasi.1,
layout=layout.fruchterman.reingold,
vertex.size=10,
vertex.label.cex=.5,
edge.arrow.size=.5,
mark.groups=list(members),
mark.col="green"
)
```



A preferential attachment model network with a highlighted walktrap community.

```
# With the above plot the group with the green polygon surrounding it is the nodes
# Now we will play around with the "power" argument to see how that impacts the grap
# We will generate 4 networks with preferential attachment at varying levels.
barabasi.game.2<-barabasi.game(n=50,power=.75)
barabasi.game.3<-barabasi.game(n=50,power=.5)
barabasi.game.4<-barabasi.game(n=50,power=.25)
barabasi.game.5<-barabasi.game(n=50,power=0)

# These can be organized into a list for convenience.
barabasi.graphs<-list(barabasi.game.2,barabasi.game.3,barabasi.game.4,barabasi.game.

# Now lets use community detection, this time with the walktrap algorithm.
bg.community.list<-lapply(barabasi.graphs,walktrap.community)
bg.membership.list<-lapply(bg.community.list,membership)

txt<-c("a","b","c","d")    # vector for labeling the graphs

# Plot these four graphs in one window with:
par(mfrow=c(2,2),mar=c(.2,.2,.2,.2))
# The for loop here plots each graph in the list one by one into the window prepared
for(i in 1:4){
plot.igraph(barabasi.graphs[[i]],
layout=layout.fruchterman.reingold,
vertex.size=10,
vertex.label.cex=.5,
edge.arrow.size=.5,
mark.groups=list(bg.membership.list[[i]]),
```

```
mark.col="green",
frame=T # the frame argument plots a box around the graph
)
text(1,1,txt[i]) # calls from the vector to label the graph, adds to the
}

# Later we will look at the properties of these graphs to see exactly how they are c
```



Four preferential attachment model networks with walktrap communities highlighted

```
# Graphs can also be visualized as a matrix, the adjacency matrix. The adjacency mat

# The function get.adjacency() converts a graph into a matrix.
barabasi.adjacency<-get.adjacency(graph.barabasi.1)

# A number of other functions will use the adjacency matrix to calculate different r

# And we can get a graph from the adjacency matrix with graph.adjacency(). Often the
graph.adjacency(barabasi.adjacency)

# Similarly we can arrange the information in a number of other ways

# Edgelist
barabasi.edgelist<-get.edgelist(graph.barabasi.1)

# Adjacency list
barabasi.adjlist<-get.adjlist(graph.barabasi.1,mode="all")

# Dataframe
barabasi.data.frame<-get.data.frame(graph.barabasi.1,what="edges")
```

Share this:

Related

New section on the blog
In "Other"

Network basics with R and igraph (part II of III)
In "Rbasics"

Food web plotting in R
In "Research"

This entry was posted in Rbasics and tagged basics, code, igraph, network, R. Bookmark the permalink.

4 Responses to *Network basics with R and igraph (part I of III)*

Tony *says:*
June 12, 2013 at 10:07 AM

A very helpful post!

Reply

Jon Borrelli *says:*
June 12, 2013 at 1:30 PM

thanks!

Reply

Pingback: *Blogroll: Assembling my Network | Scientific Gems*

Sai Krishna *says:*
August 23, 2014 at 10:42 PM

Thanks a lot

Reply