Assembling my Network

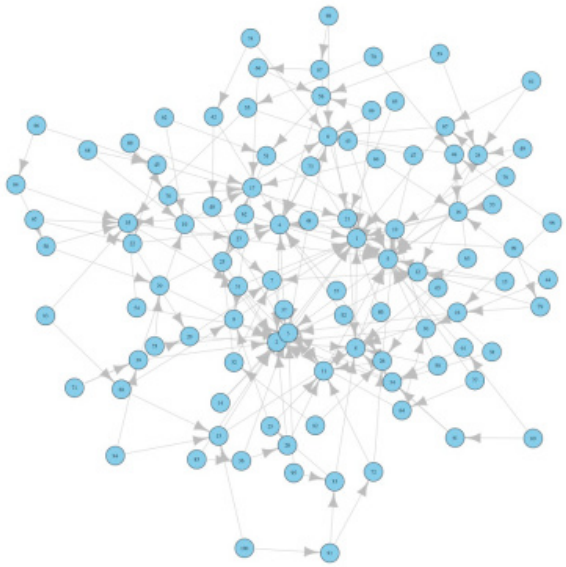
The ecology is theoretical but the fun is real.

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

Posted on June 10, 2013

The first part of this little series of posts went over a some basic network modeling and plotting tools. In this part I want to go over how to get some of the simple properties of these networks using two packages, igraph and NetIndices.

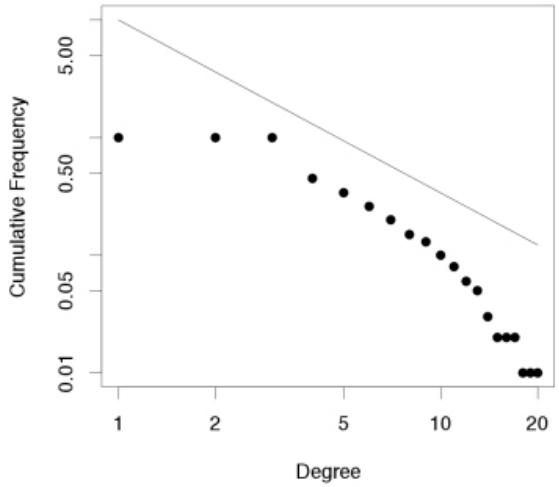
```
################################################################
################################################################
####
####            PART II: NETWORK PROPERTIES
####
################################################################
################################################################
# Again I will be using the igraph package
# to analyze various properties of networks
library(igraph)
# Additionally I will use the NetIndices package,
# since its function "GenInd()" outputs several network properties
library(NetIndices)
# First I'll create a network to analyze using
# the preferential attachment model (power=.5)

test.graph<-barabasi.game(100,power=.5,m=2)
# In this case we have set m=2, meaning that
# for each new node 2 new links are created

par(mar=c(.1,.1,.1,.1))
plot.igraph(test.graph,
layout=layout.fruchterman.reingold,
vertex.size=7,
vertex.label.cex=.5,
edge.arrow.size=.5)
```



Preferential attachment graph with m=2, power=0.5

```
# How large is the network (I know I set this when we made the network,
# but what if I had not?)

test.graph      # Tells me that it is an IGRAPH object with 100 nodes and 197 links,
# made with the Barabasi algorithm
V(test.graph)   # gives the vertex sequence
E(test.graph)   # gives the edge sequence (edge list)

# The "GenInd()" function requires an input of an adjacency matrix
test.graph.adj<-get.adjacency(test.graph,sparse=F)
# in older versions of igraph the default was sparse=F,
# but now you must specify, other wise you get a matrix of 1s and .s

test.graph.properties<-GenInd(test.graph.adj)

# The function output consists of 10 network properties.
# I will consider five of them here:

test.graph.properties$N            #number of nodes

test.graph.properties$Ltot        #number of links

test.graph.properties$LD        #link density (average # of links per node)

test.graph.properties$C            #the connectance of the graph
# This function measures connectance as L/(N*(N-1)) where L is links, and N is nodes
# Connectance can also be calculated as L/(N^2)

# The degree of a node refers to the number of links associated with a node.
# Degree can be measured as the links going in ("in degree"), out ("out degree"), or
# The degree() function takes a graph input and gives the degree of specified nodes.
# With the argument "v=V(graph)" you tell the function to give the degree of all nod
# while the "mode" argument specifies in, out, or both.

in.deg.testgraph<-degree(test.graph,v=V(test.graph),mode="in")
```

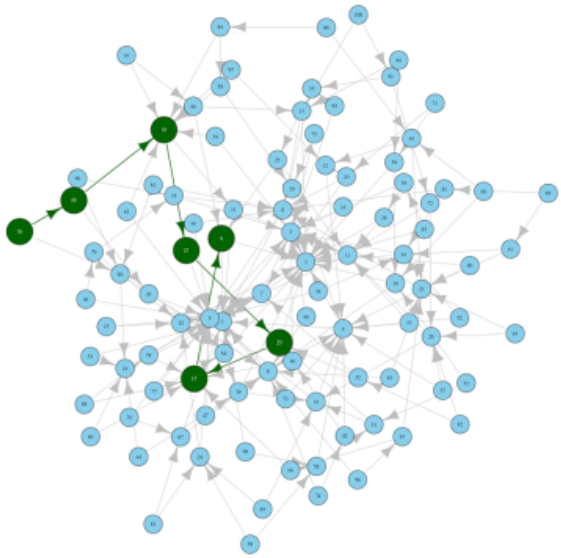
```
out.deg.testgraph<-degree(test.graph,v=V(test.graph),mode="out")
all.deg.testgraph<-degree(test.graph,v=V(test.graph),mode="all")

# Degree distribution is the cumulative frequency of nodes with a given degree
# this, like degree() can be specified as "in", "out", or "all"
deg.distr<-degree.distribution(test.graph,cumulative=T,mode="all")

# Using the power.law.fit() function I can fit a power law to the degree distributi
power<-power.law.fit(all.deg.testgraph)

# The output of the power.law.fit() function tells me what the exponent of the power
# and the log-likelihood of the parameters used to fit the power law distribution ($
# Also, it performs a Kolmogov-Smirnov test to test whether the given degree distrib
# been drawn from the fitted power law distribution.
# The function thus gives me the test statistic ($KS.stat) and p-vaule ($KS.p) for t

# Then I can plot the degree distribution
plot(deg.distr,log="xy",
ylim=c(.01,10),
bg="black",pch=21,
xlab="Degree",
ylab="Cumulative Frequency")

# And the expected power law distribution
lines(1:20,10*(1:20)^((-power$alpha)+1))

# Graphs typically have a Poisson distribution (if they are random),
# power law (preferential attachment), or truncated power law (many real networks) c
```



Cumulative degree distribution with power law fitted

```
# Diameter is essentially the longest path between two vertices
diameter(test.graph)
# Gives me the length of the diameter while

```

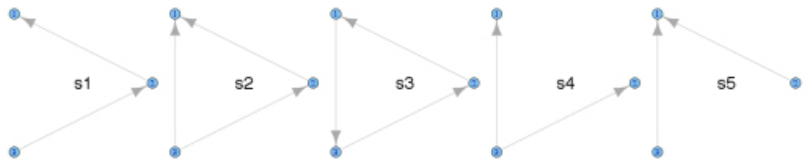
```
nodes.diameter<-get.diameter(test.graph)
# Gives me the labels for each node that participates in the diameter

# I can look at the diameter graphically also
# First I will define the node and edge attributes
V(test.graph)$color<-"skyblue"
# I want all the nodes to be skyblue
V(test.graph)$size<-7
# I want all the nodes to be size=7
V(test.graph)[nodes.diameter]$color<-"darkgreen"
V(test.graph)[nodes.diameter]$size<-10
V(test.graph)[nodes.diameter]$label.color<-"white"
# but the nodes in the diameter should be darkgreen and larger than the rest
# with a white label instead of black
# this will make the diameter pop out of the larger network
E(test.graph)$color<-"grey"
# all non-diameter edges will be grey
E(test.graph,path=nodes.diameter)$color<-"darkgreen"
E(test.graph,path=nodes.diameter)$width<-2
# Edges in the diameter will be darkgreen and a little extra wide

# If you do not set the attributes of all of the nodes and edges then it will
# default such that you only see what you have defined

# Now when I plot the diameter will be larger than everything else, and darkgreen ir
# of grey/blue
par(mar=c(.1,.1,.1,.1))
plot.igraph(test.graph,
layout=layout.fruchterman.reingold,
vertex.label.cex=.5,
edge.arrow.size=.5)
```



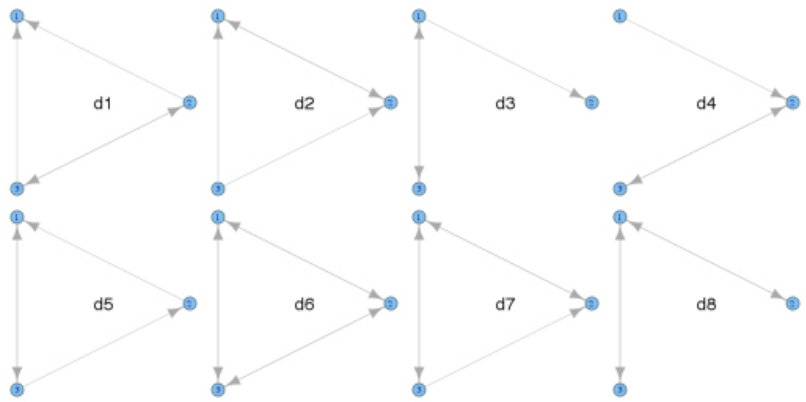
test.graph with diameter highlighted

```
# Clustering coefficient is the proportion of
# a nodes neighbors that can be reached by other neighbors
# in igraph this property is apparently called "transitivity"
```

```

transitivity(test.graph)
# gives the clustering coefficient of the whole network

transitivity(test.graph,type="local")
# gives the clustering coefficient of each node

# Betweenness is the number of shortest paths between two nodes that go through each

graph.betweenness<-betweenness(test.graph,v=V(test.graph))
graph.edge.betweenness<-edge.betweenness(test.graph,e=E(test.graph))

# Closeness refers to how connected a node is to its neighbors

graph.closeness<-closeness(test.graph,vids=V(test.graph))

# Clustering coefficient, betweenness, and closeness
# all describe the small world properties of the network.
# A network with small world properties is one in which
# it takes a relatively short path to get from one node to the next
# (e.g., six degrees of separation)

# Every graph can be decomposed into its component n-node subgraphs.
# In particular there are 13 unique ways to arrange 3 nodes in directed graphs.
# Here are the adjacency matrices for each of the 13 subgraphs
s1<-matrix(c(0,1,0,0,0,1,0,0,0),nrow=3,ncol=3)
s2<-matrix(c(0,1,1,0,0,1,0,0,0),nrow=3,ncol=3)
s3<-matrix(c(0,1,0,0,0,1,1,0,0),nrow=3,ncol=3)
s4<-matrix(c(0,0,1,0,0,1,0,0,0),nrow=3,ncol=3)
s5<-matrix(c(0,1,1,0,0,0,0,0,0),nrow=3,ncol=3)
d2<-matrix(c(0,1,1,1,0,1,0,0,0),nrow=3,ncol=3)
d1<-matrix(c(0,1,1,0,0,1,0,1,0),nrow=3,ncol=3)
d3<-matrix(c(0,0,1,1,0,0,1,0,0),nrow=3,ncol=3)
d4<-matrix(c(0,0,0,1,0,1,0,1,0),nrow=3,ncol=3)
d5<-matrix(c(0,1,1,0,0,1,1,0,0),nrow=3,ncol=3)
d6<-matrix(c(0,1,1,1,0,1,1,1,0),nrow=3,ncol=3)
d7<-matrix(c(0,1,1,1,0,1,1,0,0),nrow=3,ncol=3)
d8<-matrix(c(0,1,1,1,0,0,1,0,0),nrow=3,ncol=3)

# I then make the 13 matrices into a list
subgraph3.mat<-list(s1,s2,s3,s4,s5,d1,d2,d3,d4,d5,d6,d7,d8)
# And convert the matrices into graph objects
subgraph3.graph<-lapply(subgraph3.mat,graph.adjacency)

# Here I have created a simple for loop to go through the list of subgraphs
# and count how many times that subgraph appears in the larger test.graph
subgraph.count<-c()
for(i in 1:13){
subgraph.count[i]<-
graph.count.subisomorphisms.vf2(test.graph,subgraph3.graph[[i]])
}
```



3 node subgraphs with only single links considered



3 node subgraphs with double links considered

Related

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

New section on the blog
In "Other"

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

This entry was posted in Rbasics and tagged Barabási, code, indices, network, R, statistics. Bookmark the permalink.

8 Responses to *Network basics with R and igraph (part II of III)*

Tony *says:*

June 12, 2013 at 10:08 AM

Part 3 still to come?

Reply

Jon Borrelli *says:*

June 12, 2013 at 10:15 AM

soon, hopefully!

Reply

Luiz Felipe Freitas *says:*

June 19, 2013 at 12:40 PM

Thank you very much! Those two posts have been very helpful.
Looking forward to Part 3.

Reply

Jon Borrelli *says:*

June 19, 2013 at 1:48 PM

I am glad you found them helpful! I am planning the third part for sometime next week, and it will be focused on getting information out of a food web dataset, so it will be less of a general network overview and more specific to ecology.

Reply

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

swarup *says:*

July 17, 2013 at 6:45 AM

heii in my system degree() is not working.....
can someone help me?

it is giving this error :
> in.deg.testgraph<-degree(test.graph,v=V(test.graph),mode="in")
Error in degree(test.graph, v = V(test.graph), mode = "in") :
unused arguments (v = V(test.graph), mode = "in")

Reply

Jon Borrelli *says:*

July 18, 2013 at 4:40 PM

I am not sure what the problem is here, it looks like the code is correct. Does degree(test.graph) work properly?

Reply

swarup *says:*
July 31, 2013 at 1:55 AM

Hi Jon,
The problem solved. degree(test.graph) work properly. Actually there was a standard bug corresponding to degree.distribution() function in igraph. I have used $density instead of $intensities inside the function and then your code runs smoothly.
Thanks.
