STA6800: Week04 - Mathematical Models for Network Graphs

Ick Hoon Jin

2021-08-10

Introduction

1. By a model for a network graph we mean effectively a collection

$$\{P_\theta(G), G \in \mathcal{G} : \theta \in \Theta\},$$

where G is a collection (or 'ensemble') of possible graphs, P_θ is a probability distribution on $\mathcal{G}$, and θ is a vector of parameters, ranging over possible values in Θ. When convenient, we may often drop the explicit reference to θ and simply write P for the probability function.

2. Variety of Purposes

- The testing for 'significance' of a pre-defined characteristic(s) in a given network graph
- The study of proposed mechanisms for generating certain commonly observed properties in real-world networks (such as broad degree distributions or small-world effects),
- The assessment of potential predictive factors of relational ties.

3. The richness of network graph modeling derives largely from how we choose to specify $P(\cdot)$, with methods in the literature ranging from the simple to the complex.

4. It is useful for our purposes to distinguish, broadly speaking, between models defined more from (i) a mathematical perspective, versus (ii) a statistical perspective.

- Those of the former class tend to be simpler in nature and more amendable to mathematical analysis yet, at the same time, do not always necessarily lend themselves well to formal statistical techniques of model fitting and assessment.
- On the other hand, those of the latter class typically are designed to be fit to data, but their mathematical analysis can be challenging in some cases.
- Nonetheless, both classes of network graph models have their uses for analyzing network graph data.

Classical Random Graph Models

1. The term random graph model typically is used to refer to a model specifying a collection $\mathcal{G}$ and a uniform probability $P(\cdot)$ over $\mathcal{G}$. Random graph models are arguably the most well-developed class of network graph models, mathematically speaking.

2. Classical theory of random graph models by Erdos and Renyi.

- It rests upon a simple model that places equal probability on all graphs of a given order and size.
- Model specifies a collection $\mathcal{G}_{N_v,N_e}$ of all graphs $G = (V, E)$ with $|V| = N_v$ and $|E| = N_e$,and assigns probability $P(G) = \binom{N}{N_e}^{-1}$ to each $G \in \mathcal{G}_{N_v,N_e}$, where $N = \binom{N_v}{2}$ is the total number of distinct vertex pairs.

3. A variant of $\mathcal{G}_{N_v,N_e}$ arguably is seen more often in practice. In this formulation, a collection $\mathcal{G}_{N_v,p}$ is defined to consist of all graphs G of order N_v that may be obtained by assigning an edge independently to each pair of distinct vertices with probability $p \in (0, 1)$.

- This type of model sometimes is referred to as a Bernoulli random graph model.
- When p is an appropriately defined function of N_v, and $N_e \sim pN_v^2$, these two classes of models are essentially equivalent for large N_v.

4. The function `erdos.renyi.game` in igraph can be used to simulate classical random graphs of either type.

- The choice of $N_v = 100$ vertices and a probability of $p = 0.02$ of an edge between any pair of vertices.

```
library(sand)
```

```
## Loading required package: igraph
```

```
##
## Attaching package: 'igraph'
```

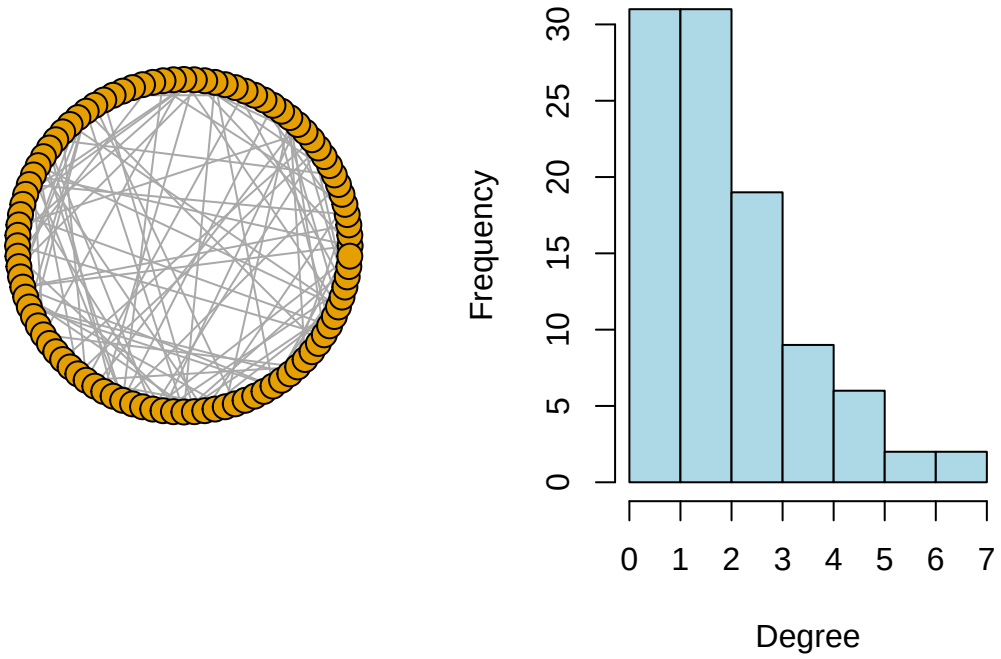
```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
## Loading required package: igraphdata
```

```
##
## Statistical Analysis of Network Data with R, 2nd Edition
## Type in C2 (+ENTER) to start with Chapter 2.
```

```
g.er = erdos.renyi.game(100, 0.02)
par(mfrow=c(1,2))
plot(g.er, layout=layout.circle, vertex.label=NA)
hist(degree(g.er), col="lightblue", xlab="Degree", ylab="Frequency", main="")
```




```
is.connected(g.er)
```

```
## [1] FALSE
```

```
table(sapply(decompose.graph(g.er),vcount))
```

```
##
##  1  2 86
## 12  1  1
```

- In general, a classical random graph G will with high probability have a giant component if $p = c/N_v$ for some $c > 1$.
- Under this same parameterization for p, for $c > 0$, the degree distribution will be well-approximated by a Poisson distribution, with mean c, for large N_v.
- That this should be true is somewhat easy to see at an intuitive level, since the degree of any given vertex is distributed as a binomial random variable, with parameters $N_v - 1$ and p.
- In our simulated random graph, the mean degree is quite close to the expected value of $(100 - 1) \times 0.02 = 1.98$.
- Other properties of classical random graphs include that there are relatively few vertices on shortest paths between vertex pairs and that there is low clustering.

```
average.path.length(g.er)
```

```
## [1] 4.689004
```

```
diameter(g.er)
```

```
## [1] 11
```

```
transitivity(g.er)
```

```
## [1] 0.04379562
```

- It can be shown that the diameter varies like $O(\log N)$, and the clustering coefficient, like N_v^{-1}.

Generalized Random Graph Models

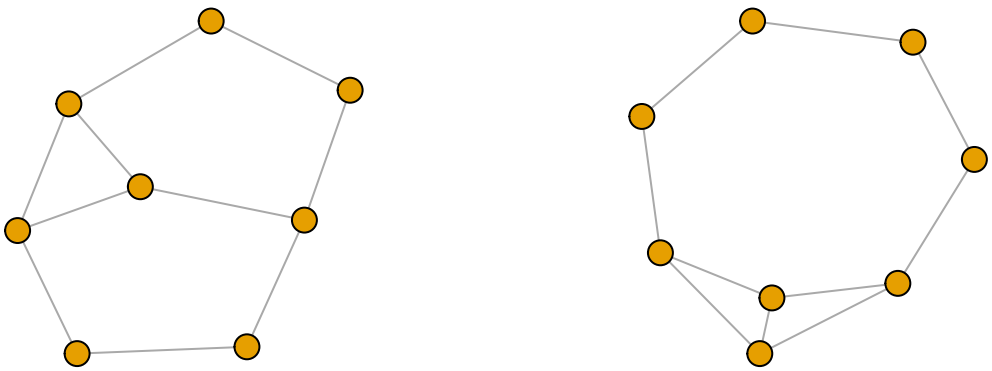
1. The formulation of Erdos and Renyi can be generalized in a straightforward manner.

- define a collection of graphs $\mathcal{G}$ consisting of all graphs of a fixed order N_v that possess a given characteristic(s)
- assign equal probability to each of the graphs $G \in \mathcal{G}$.

2. Beyond Erdos-Renyi, the most commonly chosen characteristic is that of a fixed degree sequence.

- $\mathcal{G}$ is defined to be the collection of all graphs G with a pre-specified degree sequence, which we will write here as $\{d_{(1)}, \cdots, d_{(N_v)}\}$, in ordered form.
- `degree.sequence.game` can be used to uniformly sample random graphs with fixed degree sequence.
- Suppose, for example, that we are interested in graphs of $N_v = 8$ vertices, half of which have degree $d = 2$, and the other half, degree $d = 3$.

```
degs = c(2, 2, 2, 2, 3, 3, 3, 3)
g1 = degree.sequence.game(degs, method="vl")
g2 = degree.sequence.game(degs, method="vl")
par(mfrow=c(1,2))
plot(g1, vertex.label=NA)
plot(g2, vertex.label=NA)
```



- Note that these two graphs do indeed differ, in that they are not isomorphic.

```
graph.isomorphic(g1, g2)
```

```
## [1] FALSE
```

- For a fixed number of vertices N_v, the collection of random graphs with fixed degree sequence all have the same number of edges N_e, due to the relation $\tilde{d} = 2N_e/N_v$, where $\tilde{d}$ is the mean degree of the sequence $(d_{(1)}, \cdots, d_{(N_v)})$.

```
c(ecount(g1), ecount(g2))
```

```
## [1] 10 10
```

- Therefore, this collection is strictly contained within the collection of random graphs $\mathcal{G}_{N_v,N_e}$, with the corresponding number N_v, N_e of vertices and edges. So the addition of an assumed form for the degree sequence is in this case equivalent to specifying our model through a conditional distribution on the original collection $\mathcal{G}_{N_v,N_e}$.
- On the other hand, it is important to keep in mind that all other characteristics are free to vary to the extent allowed by the chosen degree sequence.

```
data(yeast)
degs = degree(yeast)
fake.yeast = degree.sequence.game(degs, method=c("vl"))
all(degree(yeast) == degree(fake.yeast))
```

```
## [1] TRUE
```

- But the original network has twice the diameter of the simulated version, and virtually all of the substantial amount of clustering originally present is now gone.

```
diameter(yeast)
```

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

```
diameter(fake.yeast)
```

```
## [1] 8
```

```
transitivity(yeast)
```

```
## [1] 0.4686178
```

```
transitivity(fake.yeast)
```

```
## [1] 0.03958095
```

- In principle, it is easy to further constrain the definition of the class $\mathcal{G}$, so that additional characteristics beyond the degree sequence are fixed.

- Markov chain Monte Carlo (MCMC) methods are popular for generating generalized random graphs $\mathcal{G}$ from such collections, where the states visited by the Markov chain are the distinct graphs $\mathcal{G}$ themselves.

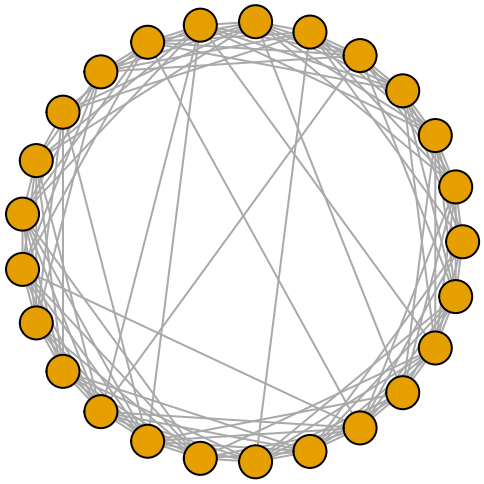
Network Graph Models Based on Mechanisms

- One of the more important innovations in modern network graph modeling is a movement from traditional random graph models to models explicitly designed to mimic certain observed 'real-world' properties, often through the incorporation of a simple mechanism(s).

Small-World Models

- A small-world network (Watts and Strogatz) is a type of mathematical graph in which most nodes are not neighbors of one another, but the neighbors of any given node are likely to be neighbors of each other and most nodes can be reached from every other node by a small number of hops or steps.
- In order to create a network graph with both of these properties, Watts and Strogatz suggested instead beginning with a graph with lattice structure, and then randomly 'rewiring' a small percentage of the edges.
  - We begin with a set of N_v vertices, arranged in a periodic faction, and join each vertex to r of its neighbors to each side.
  - For each edge, independently and with probability p, one end of that edge will be moved to be incident to another vertex, where that new vertex is chosen uniformly, but with attention to avoid the construction of loops and multi-edges.
- 'watts.strogatz.game' generates a small-world network graph.

```
g.ws = watts.strogatz.game(1, 25, 5, 0.05)
plot(g.ws, layout=layout.circle, vertex.label=NA)
```

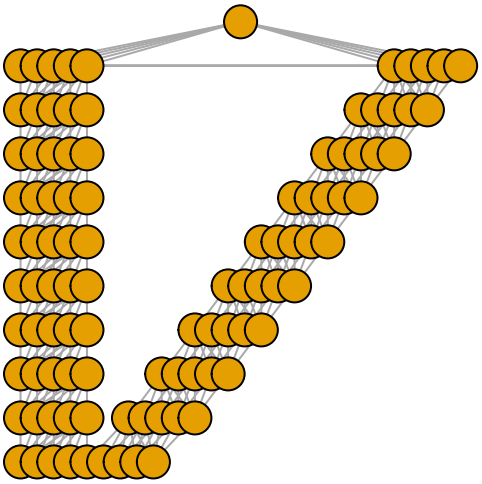


- For the lattice alone, which we generate by setting p = 0, there is a substantial amount of clustering.

```
g.lat100 = watts.strogatz.game(1, 100, 5, 0)
transitivity(g.lat100)
```

```
## [1] 0.6666667
```

```
plot(g.lat100, layout=layout.reingold.tilford, vertex.label=NA)
```



- But the distance between vertices is non-trivial.

```
diameter(g.lat100)
```

```
## [1] 10
```

```
average.path.length(g.lat100)
```

```
## [1] 5.454545
```

- The effect of rewiring a relatively small number of edges in a random fashion is to noticeably reduce the distance between vertices, while still maintaining a similarly high level of clustering. This effect may be achieved even with very small p.

```
g.ws100 = watts.strogatz.game(1, 100, 5, 0.05)
diameter(g.ws100)
```

```
## [1] 5
```

```
average.path.length(g.ws100)
```

```
## [1] 2.698788
```

```
transitivity(g.ws100)
```

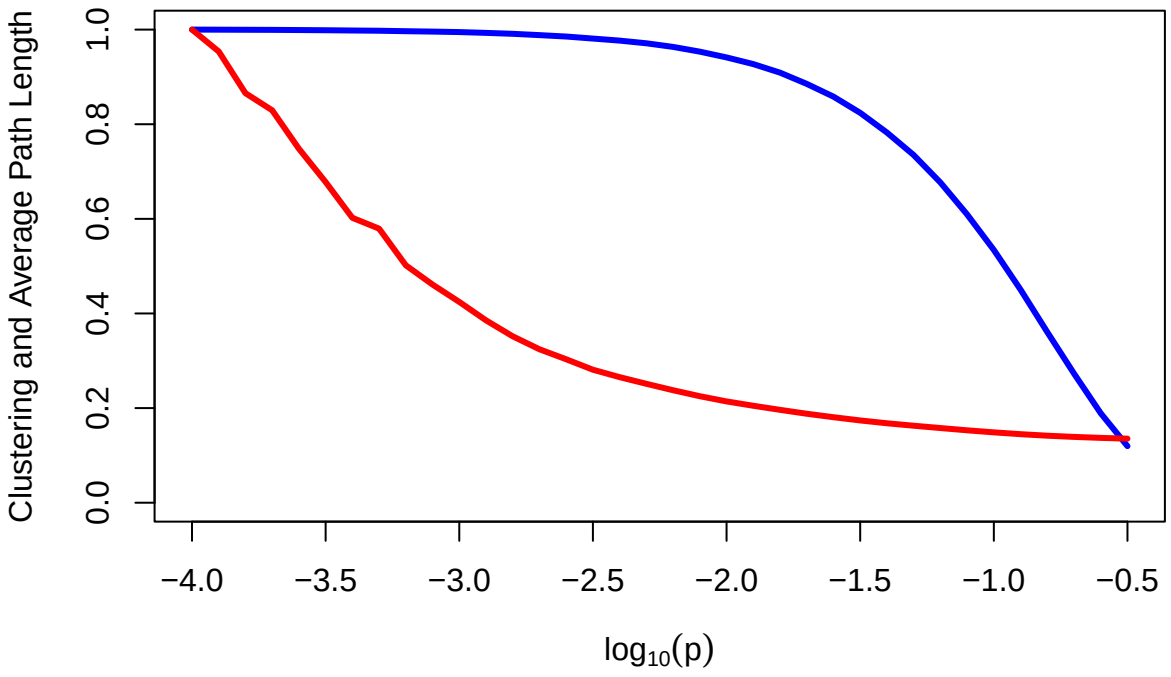
```
## [1] 0.4962654
```

- Simulate according to a particular Watts-Strogatz small-world network model, with $N_v = 1{,}000$ and $r = 10$, and re-wiring probability p, as p varies over a broad range.

```
steps = seq(-4, -0.5, 0.1)
len = length(steps)
cl = numeric(len)
apl = numeric(len)
ntrials = 100
for(i in 1:len){
  cltemp = numeric(ntrials)
  apltemp = numeric(ntrials)
  for(j in 1:ntrials){
    g = watts.strogatz.game(1, 1000, 10, 10^steps[i])
    cltemp[j] = transitivity(g)
    apltemp[j] = average.path.length(g)
  }
  cl[i] = mean(cltemp)
  apl[i] = mean(apltemp)
}
```

The results shows approximate expected values for normalized versions of average path length and clustering coefficient are plotted, indicate that over a substantial range of p the network exhibits small average distance while maintaining a high level of clustering.

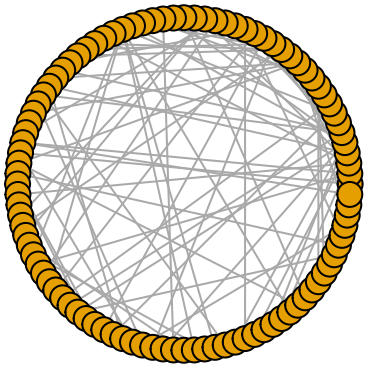
```
plot(steps, cl/max(cl), ylim=c(0,1), lwd=3, type="l", col="blue",
     xlab=expression(log[10](p)), ylab="Clustering and Average Path Length")
lines(steps, apl/max(apl), lwd=3, col="red")
```

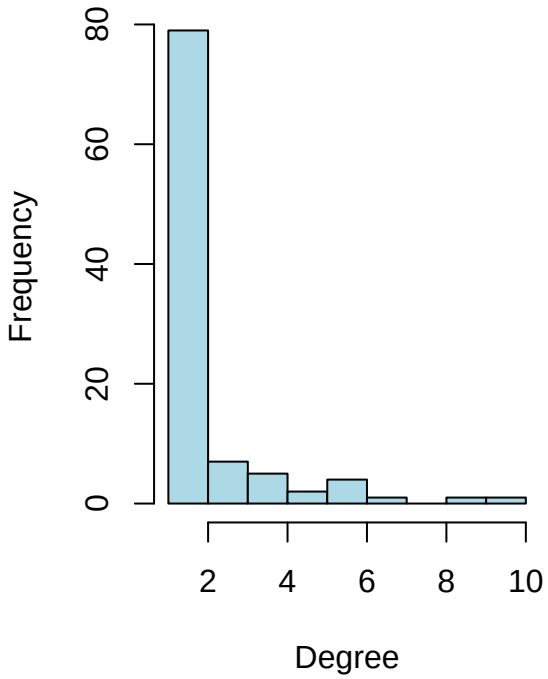


Preferential Attachment Models

- Many networks grow or otherwise evolve in time.
  - The World Wide Web and scientific citation networks are two obvious examples.
  - Similarly, many biological networks may be viewed as evolving as well, over appropriately defined time scales.
- Typically a simple mechanism(s) is specified for how the network changes at any given point in time, based on concepts like vertex preference, fitness, copying, age, and the like.
- Preferential attachment means that the more connected a node is, the more likely it is to receive new links. Nodes with a higher degree have a stronger ability to grab links added to the network. Intuitively, the preferential attachment can be understood if we think in terms of social networks connecting people.
  - It designed to embody the principle that 'the rich get richer'.
  - A driving motivation behind the introduction of this particular mechanism was a desire to reproduce the types of broad degree distributions observed in many large, real-world networks.
- The Barabasi-Albert (BA) model for undirected graphs is formulated as follows.
  1. Start with an initial graph $G^{(0)}$ of $N_v^{(0)}$ vertices and $N_e^{(0)}$ edges.
  2. Then, at stage $t = 1, 2, \cdots$, the current graph $G^{(t-1)}$ is modified to create a new graph $G^{(t)}$ by adding a new vertex of degree $m \geq 1$, where the m new edges are attached to m different vertices in $G^{(t-1)}$, and the probability that the new vertex will be connected to a given vertex v is given by
  $$\frac{d_v}{\sum_{v' \in V} d_{v'}}.$$
  That is, at each stage, m existing vertices are connected to a new vertex in a manner preferential to those with higher degrees.
  3. After t iterations, the resulting graph $G^{(t)}$ will have $N_v^{(t)} = N_v^{(0)} + t$ vertices and $N_e^{(t)} = N_e^{(0)} + tm$ edges.
- Because of the tendency towards preferential attachment, intuitively we would expect that a number of vertices of comparatively high degree should gradually emerge as t increases.
- `barabasi.game` simulate a BA random graph.

```
g.ba = barabasi.game(100, directed=FALSE)
par(mfrow=c(1,2))
plot(g.ba, layout=layout.circle, vertex.label=NA)
hist(degree(g.ba), col="lightblue", xlab="Degree", ylab="Frequency", main="")
```





- Note that the edges are spread among vertex pairs in a decidedly less uniform man- ner than in the classical random graph. And, in fact, there appear to be vertices of especially high degree—so-called 'hub' vertices.
- Examination of the degree distribution confirms this suspicion, and indicates, moreover, that the overall distribution is quite heterogeneous. Actually, the vast majority of vertices have degree no more than two in this graph.

```
summary(degree(g.ba))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    1.00    1.98    2.00   10.00
```

- The most celebrated property of such preferential attachment models is that, in the limit as t tends to infinity, the graphs $G^{(t)}$ have degree distributions that tend to a power-law form $d^{-\alpha}$, with $\alpha = 3$. This behavior is in noted contrast to the case of classical random graphs.
- On the other hand, network graphs generated according to the BA model will share with their classical counterparts the tendency towards relatively few vertices on shortest paths between vertex pairs and low clustering.

```
average.path.length(g.ba)
```

```
## [1] 5.526667
```

```
diameter(g.ba)
```

```
## [1] 13
```

```
transitivity(g.ba)
```

```
## [1] 0
```

Assessing Significance of Network Graph Characteristics

- The network graph models we have described above generally are too simple for serious statistical modeling with observed networks.
- Nonetheless, from a statistical hypothesis testing perspective, they still have a useful role to play in network analysis. Specifically, the network models introduced in this chapter are frequently used in the assessment of significance of network graph characteristics.
- Suppose that we have a graph derived from observations of some sort, which we will denote as G^{obs} here, and that we are interested in some structural characteristic, say $\eta(\cdot)$.
- In many contexts it is natural to ask whether $\eta(Gobs)$ is 'significant,' in the sense of being somehow unusual or unexpected. Network models of the type we have considered so far are used in establishing a well-defined frame of reference. That is, for a given collection of graphs G, the value $\eta(G_{obs})$ is compared to the collection of values $\{\eta(G) : G \in \mathcal{G}\}$.
- If $\eta(G_{obs})$ is judged to be extreme with respect to this collection, then that is taken as evidence that G_{obs} is unusual in having this value.
- When random graph models are used, it is straightforward to create a formal reference distribution which, under the accompanying assumption of uniform probability of elements in G, takes the form

$$P_{\eta,\mathcal{G}}(t) = \frac{\#\{G \in \mathcal{G} : \eta(G) \leq t\}}{|\mathcal{G}|}$$

  If $\eta(G^{obs})$ is found to be sufficiently unlikely this distribution, this is taken as evidence hypothesis that G^{obs} is uniform draw from $\mathcal{G}$.
- This principle lies at the heart of methods aimed at the detection of network motifs to be small subgraphs occurring far more frequently in a given network than in comparable random graphs.

Assessing the Number of Communities in a Network

- Our use of hierarchical clustering, through the function `fastgreedy.community`, resulted in the discovery of three communities in the karate network.
- In defining our frame of reference we will use two choices of G: (i) graphs of the same order $N_v = 34$ and size $N_e = 78$ as the karate network, and (ii) graphs that obey the further restriction that they have the same degree distribution as the original.
- Monte Carlo methods, using some of the same R functions encountered above, allow us to quickly generate approximations to the corresponding reference distributions. In order to do so, we need

the order, size, and degree sequence of the original karate network.

  - Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle.

```
data(karate)
nv = vcount(karate)
ne = ecount(karate)
degs = degree(karate)
ntrials = 1000
```

- We then generate classical random graphs of this same order and size and, for each one, we use the same community detection algorithm to determine the number of communities.

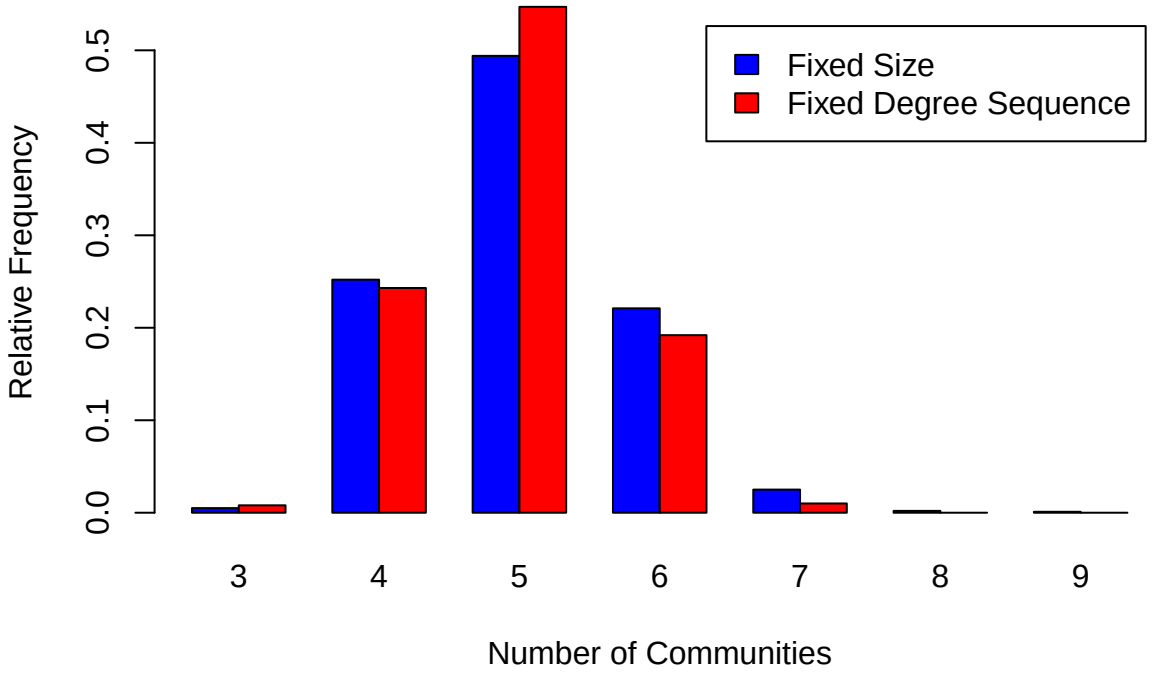
```
num.comm.rg = numeric(ntrials)
for(i in 1:ntrials){
  g.rg = erdos.renyi.game(nv, ne, type="gnm")
  c.rg = fastgreedy.community(g.rg)
  num.comm.rg[i] = length(c.rg)
}
```

- Similarly, we do the same using generalized random graphs constrained to have the required degree sequence.

```
num.comm.grg = numeric(ntrials)
for(i in 1:ntrials){
  g.grg = degree.sequence.game(degs, method="vl")
  c.grg = fastgreedy.community(g.grg)
  num.comm.grg[i] = length(c.grg)
}
```

- The results may be summarized and compared using side by side bar plots.

```
rslts = c(num.comm.rg,num.comm.grg)
indx = c(rep(0, ntrials), rep(1, ntrials))
counts = table(indx, rslts)/ntrials
barplot(counts, beside=TRUE, col=c("blue", "red"),
xlab="Number of Communities", ylab="Relative Frequency",
legend=c("Fixed Size", "Fixed Degree Sequence"))
```



- Clearly the actual number of communities detected in the original karate network (i.e., three) would be considered unusual from the perspective of random graphs of both fixed size and fixed degree sequence.
- Accordingly, we may conclude that there is likely an additional mechanism(s) at work in the actual karate club, one that goes beyond simply the density and the distribution of social interactions in this network.

Assessing Small World Properties

- A typical approach to assessing small-world behavior in this area is to compare the observed clustering coefficient and average (shortest) path length in an observed network to what might be observed in an appropriately calibrated classical random graph.
- Expect under such a comparison - if indeed an observed network exhibits small - world behavior-that the observed clustering coefficient exceed that of a random graph, while the average path length remain roughly the same.
- `macaque` contains a network of established functional connections between brain areas understood to be involved with the tactile function of the visual cortex in macaque monkeys. It is a directed network of 45 vertices and 463 links.

```
library(igraphdata)
data(macaque)
summary(macaque)
```

```
## IGRAPH f7130f3 DN-- 45 463 --
## + attr: Citation (g/c), Author (g/c), shape (v/c), name (v/c)
```

- In order to assess clustering in this network, we use an extension to directed graphs of the clustering coefficient. This quantity is defined as the average, over all vertices v, of the vertex-specific clustering coefficient

$$\mathrm{cl}(v) = \frac{\left(A + A^T\right)^3_{vv}}{2\left[d_v^{tot}\left(d_v^{tot} - 1\right) - 2\left(A^2\right)_{vv}\right]}$$

where A is the adjacency matrix and d_v^{tot} is the total degree (i.e., in-degree plus out-degree) of vertex v. The calculation of this quantity is accomplished through the following function.

```
clust.coef.dir <- function(graph){
  A = as.matrix(get.adjacency(graph))
  S = A + t(A)
  deg = degree(graph, mode=c("total"))
  num = diag(S %*% S %*% S)
  denom = diag(A %*% A)
  denom = 2 * (deg * (deg - 1) - 2 * denom)
  cl = mean(num / denom)
  return(cl)
}
```

- The steps required to simulate draws of (directed) classical random graphs, and to assess the clustering and average path length for each.

```
ntrials = 1000
nv = vcount(macaque)
ne = ecount(macaque)
cl.rg = numeric(ntrials)
apl.rg = numeric(ntrials)
for(i in 1:ntrials){
  g.rg = erdos.renyi.game(nv, ne, type="gnm", directed=TRUE)
  cl.rg[i] = clust.coef.dir(g.rg)
  apl.rg[i] = average.path.length(g.rg)
}
```

- Summarizing the resulting distributions of clustering coefficient and average path length and comparing against these distributions the values for the macaque network

```
summary(cl.rg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2166  0.2297  0.2337  0.2338  0.2375  0.2538
```

```
summary(apl.rg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.809   1.827   1.832   1.833   1.838   1.865
```

```
clust.coef.dir(macaque)
```

```
## [1] 0.5501073
```

```
average.path.length(macaque)
```

```
## [1] 2.148485
```

- Therefore, we see that on the one hand there is substantially more clustering in our network than expected from a random network. On the other hand, however, the shortest paths between vertex pairs are, on average, also noticeably longer.
- Hence, the evidence for small-world behavior in this network is not clear, with the results suggesting that the network behaves more like a lattice than a classical random graph.