# STA6800: Week02 - Descriptive Statistics of Networks

Ick Hoon Jin

2021-08-10

- In the study of a given complex system, questions of interest can often be re-phrased in a useful manner as questions regarding some aspect of the structure or characteristics of a corresponding network graph.
  - Various types of basic social dynamics can be represented by triplets of vertices with a particular pattern of ties among them (i.e., triads).
  - Questions involving the movement of information or commodities usually can be posed in terms of paths on the network graph and flows along those paths.
  - Certain notions of the 'importance' of individual system elements may be captured by measures of how 'central' the corresponding vertex is in the network.
  - The search for 'communities' and analogous types of unspecified 'groups' within a system frequently may be addressed as a graph partitioning problem.
- The structural analysis of network graphs has traditionally been treated primarily as a descriptive task, as opposed to an inferential task, and the tools commonly used for such purposes derive largely from areas outside of 'mainstream' statistics.
  - An overwhelming proportion of these tools are naturally graph-theoretic in nature, and thus have their origins in mathematics and computer science.
  - The field of social network analysis has been another key source, contributing tools usually aimed at least originally at capturing basic aspects of social structure and dynamics.
  - The field of physics has also been an important contributor, with the proposed tools often motivated by analogues in statistical mechanics.

## Vertex and Edge Characteristics

- The fundamental elements of network graphs are their vertices and edges
- Characterization of the Vertex and Edges
  - Characterizations based upon vertex degrees
  - Characterizations seeking to capture some more general notion of the 'importance' of a vertex

## Vertex Degree

- The degree $d_v$ of a vertex $v$, in a network graph $G = (V, E)$, counts the number of edges in $E$ incident upon $v$.
- Given a network graph $G$, define $f_d$ to be the fraction of vertices $v \in V$ with degree $d_v = d$.
- The collection $\{f_d\}_{d \geq 0}$ is called the degree distribution of $G$, and is simply a rescaling of the set of degree frequencies, formed from the original degree sequence.

Karate club network

```
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.
```

```
data(karate)
hist(degree(karate), col="lightblue", xlim=c(0,50),
     xlab="Vertex Degree", ylab="Frequency", main="")
```

- The degree distribution
  - There are three distinct groups of vertices, as measured by degree.
  - The two most highly connected vertices correspond to actors 1 and 34 in the network, representing the instructor and administrator about whom the club eventually split.
  - The next set of vertices consists of actors 2, 3, and also 33.
- Weighted Networks
  - A useful generalization of degree is the notion of vertex strength, which is obtained simply by summing up the weights of edges incident to a given vertex.
  - The distribution of strength sometimes called the weighted degree distribution is defined in analogy to the ordinary degree distribution.

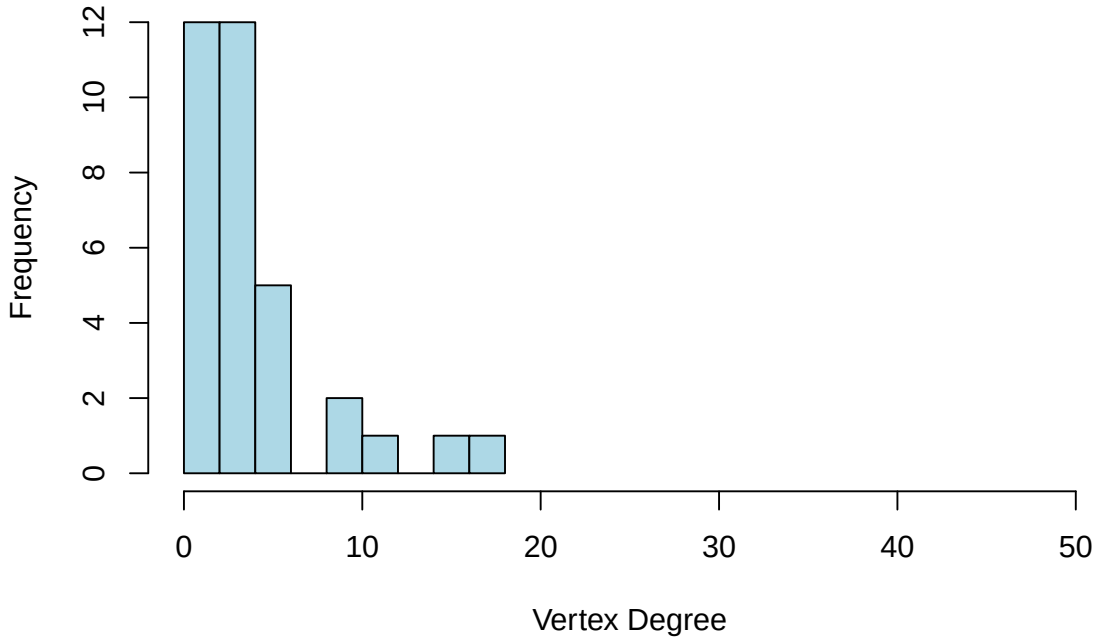


Figure 1: The degree distribution for the Karate club network

```
hist(graph.strength(karate), col="pink",
     xlab="Vertex Strength", ylab="Frequency", main="")
```

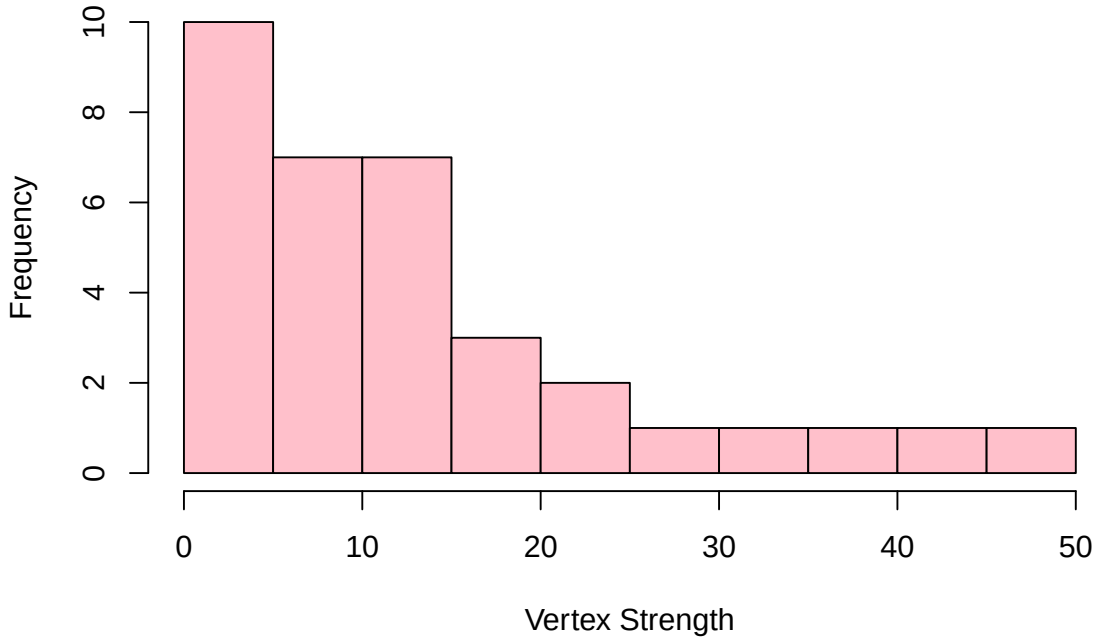


Figure 2: The vertex strength distribution for the Karate club network

A Network of Interactions among Protein Pairs in Yeast

```
library(igraphdata)
data(yeast)
ecount(yeast)
```

```
## [1] 11855
```

```
vcount(yeast)
```

```
## [1] 2617
```

- Histogram: In particular, while there is a substantial fraction of vertices of quite low degree, of an order of magnitude similar to those of the karate network, there are also a non-trivial number of vertices with degrees at successively higher orders of magnitude.
- There is a fairly linear decay in the log-frequency as a function of log-degree.

```
par(mfrow=c(1,2))
d.yeast = degree(yeast)
hist(d.yeast,col="blue", xlab="Degree", ylab="Frequency", main="Degree Distribution")
dd.yeast = degree.distribution(yeast)
d = 1:max(d.yeast)-1
ind = (dd.yeast != 0)
plot(d[ind], dd.yeast[ind], log="xy", col="blue", xlab=c("Log-Degree"),
     ylab=c("Log-Intensity"), main="Log-Log Degree Distribution")
```

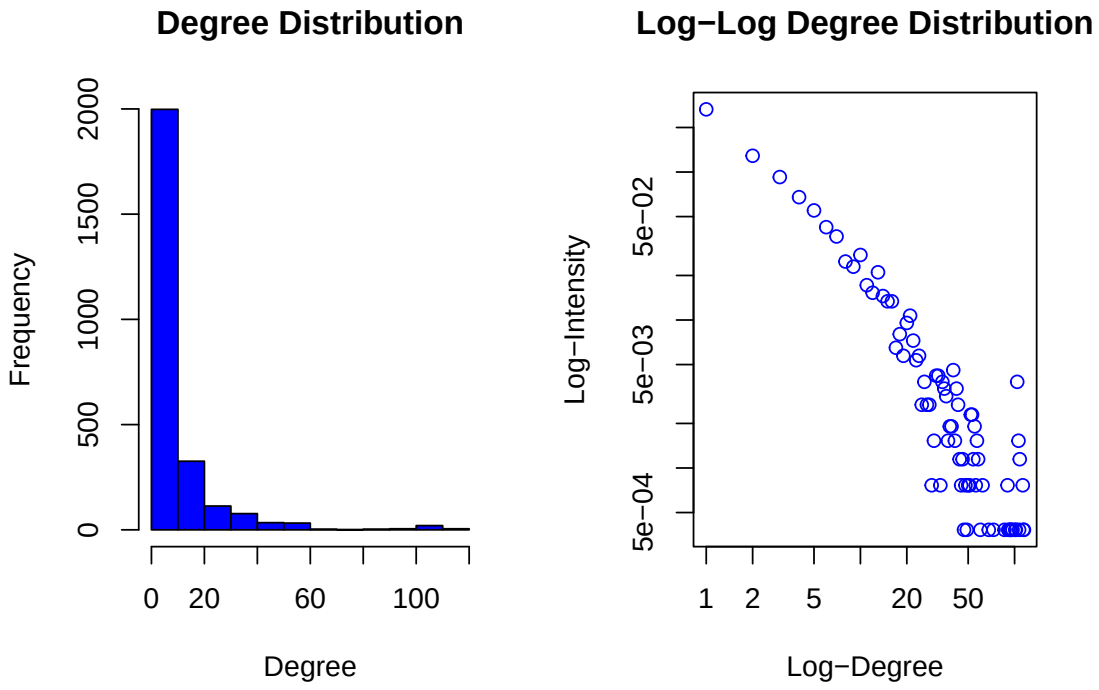


Figure 3: The degree distribution for protein interactions in Yeast

- It can be interesting to understand the manner in which vertices of different degrees are linked with each other.
- Useful in assessing this characteristic is the notion of the average degree of the *neighbors* of a given vertex.
- While there is a tendency for vertices of higher degrees to link with similar vertices, vertices of lower degree tend to link with vertices of both lower and higher degrees.

```
a.nn.deg.yeast = graph.knn(yeast,V(yeast))$knn
plot(d.yeast, a.nn.deg.yeast, log="xy", col="goldenrod",
     xlab=c("Log Vertex Degree"), ylab=c("Log Average Neighbor Degree"))
```

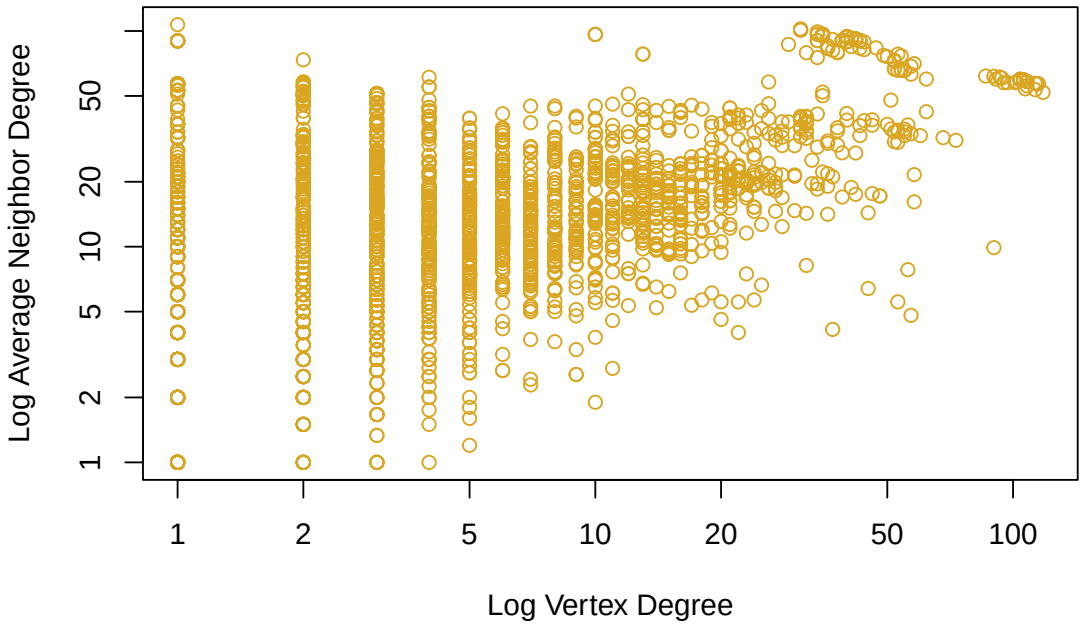


Figure 4: The degree distribution for protein interactions in Yeast

## Vertex Centrality

- Many questions that might be asked about a vertex in a network graph essentially seek to understand its 'importance' in the network.
  - Which actors in a social network seem to hold the 'reins of power'?
  - How authoritative does a particular page in the World Wide Web seem to be considered?
  - The deletion of which genes in a gene regulatory network is likely to be lethal to the corresponding organism?
  - How critical is a given router in an Internet network to the flow of traffic?
- Measures of centrality are designed to quantify such notions of 'importance' and thereby facilitate the answering of such questions.
- Most widely used measure of vertex centrality: Vertex Degree.
- Three other classic types of vertex centrality measures: Closeness, Betweenness, and Eigenvector centrality
- *Closeness* centrality measures attempt to capture the notion that a vertex is 'central' if it is 'close' to many other vertices.
  - The standard approach is to let the centrality vary inversely with a measure of the total

distance of a vertex from all others,

$$c_{CL} = \frac{1}{\sum_{u \in V} \text{dist}(u, v)},$$

where $\text{dist}(v, u)$ is the geodesic distance between the vertices $u, v \in V$.

- Often, for comparison across graphs and with other centrality measures, this measure is normalized to lie in the interval $[0, 1]$, through multiplication by a factor $N_v 1$.

- *Betweenness centrality* measures are aimed at summarizing the extent to which a vertex is located 'between' other pairs of vertices.
  - These centralities are based upon the perspective that 'importance' relates to where a vertex is located with respect to the paths in the network graph.
  - If we picture those paths as the routes by which communication of some sort or another takes place, vertices that sit on many paths are likely more critical to the communication process.
  - The most commonly used betweenness centrality is defined as

$$C_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s, t \mid v)}{\sigma(s, t)}$$

    where $\sigma(s, t \mid v)$ is the total number of shortest paths between $s$ and $t$ that pass through $v$, and $\sigma(s, t)$ is the total number of shortest paths between $S$ and $t$ (regardless of whether or not they pass through $v$).
  - This centrality measure can be restricted to the unit interval through division by a factor of $(N_v - 1)(N_v - 2)/2$.

- Other centrality measures are based on notions of 'status' or 'prestige' or 'rank.'
  - Seek to capture the idea that the more central the neighbors of a vertex are, the more central that vertex itself is.
  - These measures are inherently implicit in their definition and typically can be expressed in terms of eigenvector solutions of appropriately defined linear systems of equations.
  - There are many such eigenvector centrality measures. For example,

$$C_{E_i}(v) = \alpha \sum_{\{u,v\} \in E} c_{E_i}(u).$$

    The vector $C_{E_i} = (C_{E_i}(1), \cdots, C_{E_i}(N_v))^T$ is the solution to the eigenvalue problem $A_{C_{E_i}}$, where $A$ is the adjacency matrix for the network graph $G$.
  - An optimal choice of $\alpha^{-1}$ is the largest eigenvalue of $A$, and hence $c_{E_i}$ is the corresponding eigenvector.
  - When $G$ is undirected and connected, the largest eigenvalue of $A$ will be simple and its eigenvector will have entries that are all nonzero and share the same sign.
  - Convention is to report the absolute values of these entries, which will automatically lie between

0 and 1 by the orthonormality of eigenvectors.

- An intuitively appealing way of displaying vertex centralities (for networks of small to moderate size) is to use a radial layout, with more central vertices located closer to the center.

```
#install.packages("network", repos="http://cran.us.r-project.org")
#install.packages("sna", repos="http://cran.us.r-project.org")
A = get.adjacency(karate, sparse=FALSE)
library(network)
```

```
##
## 'network' 1.17.1 (2021-06-12), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information
```

```
##
## Attaching package: 'network'
```

```
## The following objects are masked from 'package:igraph':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute
```

```
g = as.network.matrix(A)
library(sna)
```

```
## Loading required package: statnet.common
```

```
##
## Attaching package: 'statnet.common'
```

```
## The following objects are masked from 'package:base':
##
##     attr, order
```

```
## sna: Tools for Social Network Analysis
## Version 2.6 created on 2020-10-5.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.
```

```
##
## Attaching package: 'sna'
```

```
## The following objects are masked from 'package:igraph':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census
```

```
par(mfrow=c(1,2))
gplot.target(g, degree(g), main="Degree", circ.lab = FALSE, circ.col = "skyblue",
             usearrows = FALSE, vertex.col=c("blue", rep("red", 32), "yellow"),
             edge.col="darkgray")
gplot.target(g, closeness(g), main="Closeness", circ.lab = FALSE, circ.col = "skyblue",
             usearrows = FALSE, vertex.col=c("blue", rep("red", 32), "yellow"),
             edge.col="darkgray")
```

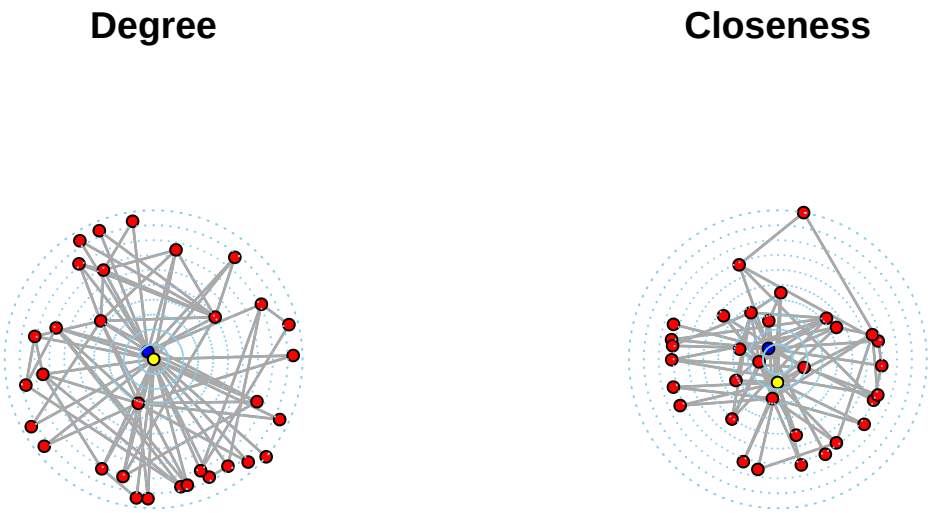


Figure 5: Target plots showing various vertex centralities for the karate club network

```
#install.packages("network", repos="http://cran.us.r-project.org")
#install.packages("sna", repos="http://cran.us.r-project.org")
A = get.adjacency(karate, sparse=FALSE)
library(network)
g = as.network.matrix(A)
library(sna)
par(mfrow=c(1,2))
gplot.target(g, betweenness(g), main="Betweenness", circ.lab = FALSE, circ.col = "skyblue",
             usearrows = FALSE, vertex.col=c("blue", rep("red", 32), "yellow"),
             edge.col="darkgray")
gplot.target(g, evcent(g), main="Eigenvalue", circ.lab = FALSE, circ.col = "skyblue",
             usearrows = FALSE, vertex.col=c("blue", rep("red", 32), "yellow"),
             edge.col="darkgray")
```

- Extensions of these centrality measures from undirected to directed graphs are straightforward.

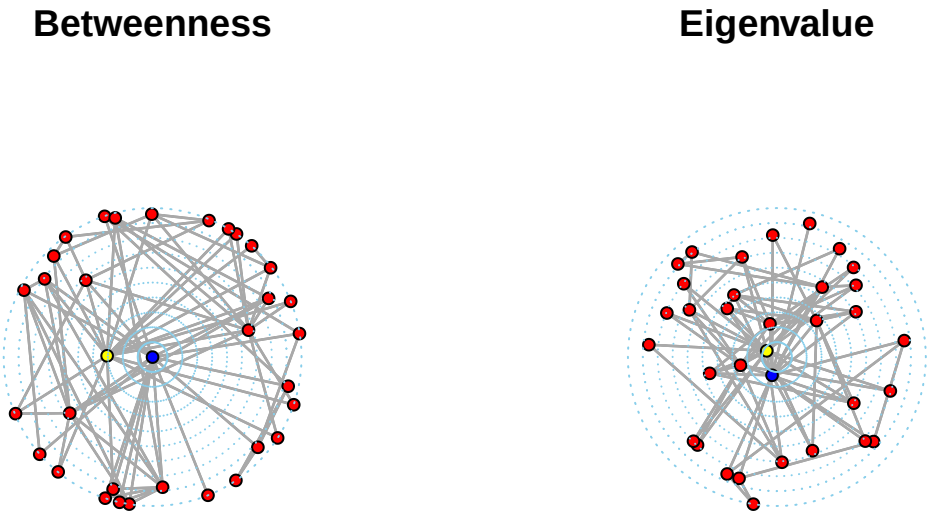


Figure 6: Target plots showing various vertex centralities for the karate club network

– Characterizes the importance of so-called hub vertices by how many authority vertices they point to, and so-called authority vertices by how many hubs point to them.
– Given an adjacency matrix $A$ for a directed graph, hubs are determined according to the eigenvector centrality of the matrix $M_{hub} = AA^T$ , and authorities, according to that of $M_{auth} = A^T A$.

```
l = layout.kamada.kawai(aidsblog)
par(mfrow=c(1,2))
plot(aidsblog, layout=l, main="Hubs", vertex.label="",
     vertex.size=10 * sqrt(hub.score(aidsblog)$vector))
plot(aidsblog, layout=l, main="Authorities", vertex.label="",
     vertex.size=10 * sqrt(authority.score(aidsblog)$vector))
```

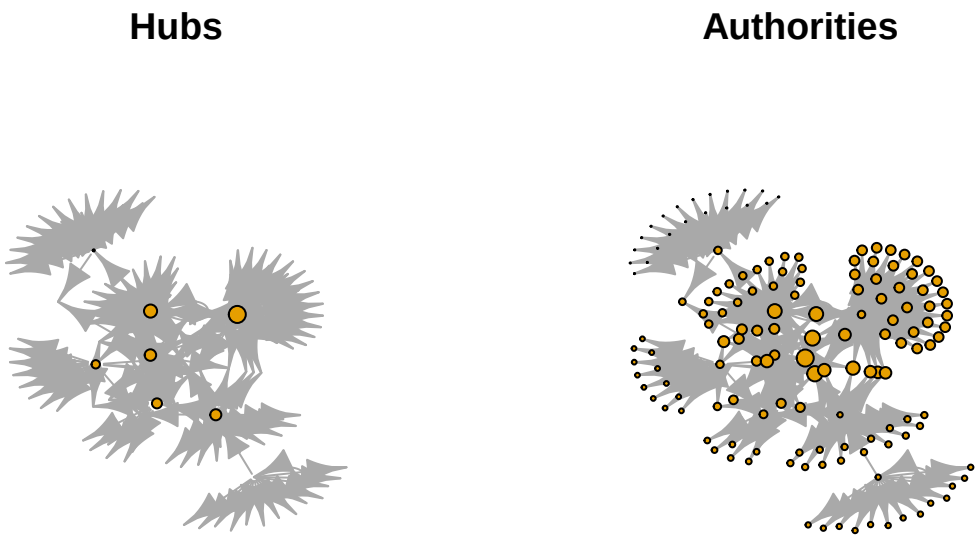


Figure 7: AIDS blog network with vertex area proportional to hubs and authority centrality measures

### Characterizing Edges

- Edge betweenness centrality which extends vertex betweenness centrality in a straightforward manner, by assigning to each edge a value that reflects the number of shortest paths traversing that edge is a natural quantity to use.
- Using edge betweenness with the karate network and examining, for instance, the edges with the three largest betweenness values

```
eb = edge.betweenness(karate)
E(karate)[order(eb, decreasing=T)[1:3]]
```

```
## + 3/78 edges from 4b458a1 (vertex names):
## [1] Actor 20--John A   Mr Hi   --Actor 20 Mr Hi   --Actor 32
```

- Many other vertex centrality measures do not extend as easily. One way around this problem is to apply vertex centrality measures to the vertices in the line graph of a network graph $G$.
- Line graph of $G$, say $G' = (V', E')$, is obtained essentially by changing vertices of $G$ to edges, and edges, to vertices.
- The vertices $v' \in V'$ represent the original edges $e \in E$, and the edges $e' \in E'$ indicate that the two corresponding original edges in $G$ were incident to a common vertex in $G$.

## Characterizing Network Cohesion

- Questions involving network cohesion, the extent to which subsets of vertices are cohesive or 'stuck together' with respect to the relation defining edges in the network graph.
  - Do friends of a given actor in a social network tend to be friends of one another as well?
  - What collections of proteins in a cell appear to work closely together?
  - Does the structure of the pages in the World Wide Web tend to separate with respect to distinct types of content?
  - What portion of a measured Internet topology would seem to constitute the 'backbone'?
- There are many ways that we can define network cohesion, depending on the context of the question being asked.
  - Definitions differ, for example, in scale, ranging from local (e.g., triads) to global (e.g., giant components), and also in the extent to which they are specified explicitly (e.g., cliques) versus implicitly (e.g., 'clusters' or 'communities').

### Subgraphs and Censuses

Cliques

- Cliques are complete subgraphs and hence are subsets of vertices that are fully cohesive, in the sense that all vertices within the subset are connected by edges.
- A census of cliques of all size can provide some sense of a 'snapshot' of how structured a graph is

```
table(sapply(cliques(karate), length))
```

```
##
##  1  2  3  4  5
## 34 78 45 11  2
```

- For the karate network a census of this sort reflects that there are 34 nodes (cliques of size one) and 78 edges (cliques of size two), followed by 45 triangles (cliques of size three).
- The largest cliques are of size five, of which there are only two.

```
cliques(karate)[sapply(cliques(karate), length) == 5]
```

```
## [[1]]
## + 5/34 vertices, named, from 4b458a1:
## [1] Mr Hi    Actor 2  Actor 3  Actor 4  Actor 14
##
## [[2]]
## + 5/34 vertices, named, from 4b458a1:
## [1] Mr Hi   Actor 2 Actor 3 Actor 4 Actor 8
```

- The cliques of larger sizes necessarily include cliques of smaller sizes.
- A maximal clique is a clique that is not a subset of a larger clique.

```
table(sapply(maximal.cliques(karate), length))
```

```
##
##  2  3  4  5
## 11 21  2  2
```

- Large cliques are relatively rare, as they necessarily require that a graph $G$ itself be fairly dense, while real-world networks are often sparse.

```
clique.number(yeast)
```

```
## [1] 23
```

Weakened Notions of Cliques

- A $k$-core of a graph $G$ is a subgraph of $G$ for which all vertex degrees are at least $k$, and such that no other subgraph obeying the same condition contains it (i.e., it is maximal in this property).
- The notion of cores is particularly popular in visualization, as it provides a way of decomposing a network into 'layers'.

```
cores = graph.coreness(karate)
gplot.target(g, cores, circ.lab = FALSE, circ.col="skyblue",
             usearrows = FALSE, vertex.col=cores, edge.col="darkgray")
```

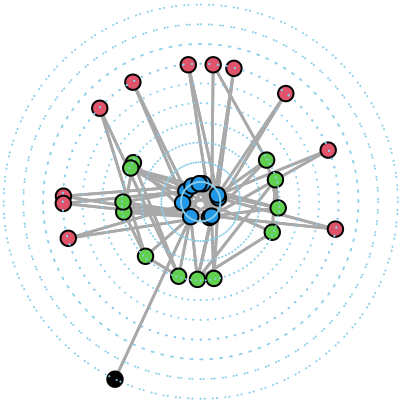

Figure 8: Visual representation of the k-core decomposition of the karate network

```
detach("package:sna")
detach("package:network")
```

Vertices of coreness one (black), two (red), three (green), and four (blue) are shown at successively smaller distances from the center, with the same distance for vertices within each core

Other Classes of Subgraphs in Defining Network Cohesion.

- Dyads are pairs of vertices and, in directed graphs, may take on three possible states: null (no directed edges), asymmetric (one directed edge), or mutual (two directed edges).
- Triads are triples of vertices and may take on 16 possible states, ranging from the null subgraph to the subgraph in which all three dyads formed by the vertices in the triad have mutual directed edges.

The vast majority of the dyads are null and, of those that are non-null, almost all are asymmetric, indicating a decided one-sidedness to the manner in which blogs in this network reference each other.

```
aidsblog = simplify(aidsblog)
dyad.census(aidsblog)
```

```
## $mut
## [1] 3
##
## $asym
## [1] 177
##
## $null
## [1] 10405
```

- Consistent with the observations from our earlier analysis of hubs and authorities in this network
- Small connected subgraphs of interest are commonly termed motifs.
- The notion of motifs is particularly popular in the study of biological networks, where arguments

often are made linking such network substructures to biological function.

## Density and Related Notions of Relative Frequency

Density

- The density of a graph is the frequency of realized edges relative to potential edges. For example, in a (undirected) graph $G$ with no self-loops and no multiple edges, the density of a subgraph $H = (V_H, E_H)$ is

$$\text{den}(H) = \frac{|E_H|}{|V_H|(|V_H| - 1)/2}$$

  The value of $\text{den}(H)$ will lie between zero and one and provides a measure of how close $H$ is to being a clique. In the case that $G$ is a directed graph, the denominator is replaced by $|V_H|(|V_H|1)$.

- Taking $H = G$ yields the density of the overall graph $G$.

- Conversely, taking $H = H_v$ to be the set of neighbors of a vertex $v \in V$, and the edges between them, yields a measure of density in the immediate neighborhood of $v$.

- The subgraphs corresponding to each of the instructor and the administrator, in union with their immediate respective neighborhoods i.e., the ego-centric networks around vertices 1 and 34 are noticeably more dense than the overall network.

```
ego.instr = induced.subgraph(karate,neighborhood(karate, 1, 1)[[1]])
ego.admin = induced.subgraph(karate,neighborhood(karate, 1, 34)[[1]])
graph.density(karate)
```

```
## [1] 0.1390374
```

```
graph.density(ego.instr)
```

```
## [1] 0.25
```

```
graph.density(ego.admin)
```

```
## [1] 0.2091503
```

Clustering Coefficients

- The standard use of the term clustering coefficient typically refers to the quantity

$$\text{cl}_T(G) = \frac{3\tau_\Delta(G)}{\tau_3(G)},$$

  where $\tau_\Delta(G)$ is the number of triangles in the graph $G$, and $\tau_3(G)$, the number of connected triples (i.e., a subgraph of three vertices connected by two edges, also sometimes called a 2-star).

- The value $\text{cl}_T(G)$ is alternatively called the transitivity of the graph, and is a standard quantity of interest in the social network literature, where it is also referred to as the 'fraction of transitive triples'.

- $\text{cl}_T(G)$ is a measure of global clustering, summarizing the relative frequency with which connected triples close to form triangles.

```
transitivity(karate)
```

```
## [1] 0.2556818
```

```
transitivity(karate, "local", vids=c(1,34))
```

```
## [1] 0.1500000 0.1102941
```

Reciprocity

- A concept unique to directed graphs
- In the case that dyads are used as units, reciprocity is defined to be the number of dyads with reciprocated (i.e., mutual) directed edges divided by the number of dyads with a single, unreciprocated edge.
- Reciprocity is defined as the total number of reciprocated edges divided by the total number of edges.

```
reciprocity(aidsblog, mode="default")
```

```
## [1] 0.03278689
```

```
reciprocity(aidsblog, mode="ratio")
```

```
## [1] 0.01666667
```

## Connectivity, Cuts, and Flows

- A basic question of interest is whether a given graph separates into distinct subgraphs. If it does not, we might seek to quantify how close to being able to do so it is.
- A graph $G$ is said to be connected if every vertex is reachable from every other (i.e., if for any two vertices, there exists a walk between the two), and that a connected component of a graph is a maximally connected subgraph.
- Often it is the case that one of the connected components in a graph $G$ dominates the others in magnitude, in that it contains the vast majority of the vertices in $G$.
- Such a component is called, borrowing terminology from random graph theory, the giant component.

```
is.connected(yeast)
```

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

- A census of all connected components within this graph, however, shows that there clearly is a giant component.

```
comps = decompose.graph(yeast)
table(sapply(comps, vcount))
```

```
##
##    2    3    4    5    6    7 2375
##   63   13    5    6    1    3    1
```

This single component contains $2375/2617 \approx 90\%$ of the vertices in the network. In contrast, none of the other components alone contain even 1%.

```
yeast.gc = decompose.graph(yeast)[[1]]
```

- A celebrated characteristic observed in the giant component of many real-world networks is the so-called small world property,
  - the shortest-path distance between pairs of vertices is generally quite small
  - the clustering is relatively high. In our network of protein– protein interactions in yeast.

```
average.path.length(yeast.gc)
```

```
## [1] 5.09597
```

```
diameter(yeast.gc)
```

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

- The shortest-path distance in this network scales more like $\log N_v$ rather than $N_v$, and therefore is considered 'small'. At the same time, the clustering in this network is relatively large

```
transitivity(yeast.gc)
```

```
## [1] 0.4686663
```

indicating that close to 50 % of connected triples close to form triangles.

Connectivity

- A graph $G$ is called $k$-vertex-connected if
  - the number of vertices $N_v > k$, and
  - the removal of any subset of vertices $X \subset V$ of cardinality $|X| < k$ leaves a subgraph that is connected.
- $G$ is called k-edge-connected if
  - $N_v \geq 2$, and
  - the removal of any subset of edges $Y \subseteq E$ of cardinality $|Y| < k$ leaves a subgraph that is connected.
- The vertex (edge) connectivity of $G$ is the largest integer such that G is k-vertex(k-edge-) connected.
- It can be shown that the vertex connectivity is bounded above by the edge connectivity, which in turn is bounded above by the minimum degree dmin among vertices in $G$.

```
vertex.connectivity(yeast.gc)
```

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

```
edge.connectivity(yeast.gc)
```

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

Thus it requires the removal of only a single well-chosen vertex or edge in order to break this subgraph into additional components.

Cut

- If the removal of a particular set of vertices (edges) in a graph disconnects the graph, that set is called a vertex-cut (edge-cut).
- A single vertex that disconnects the graph is called a cut vertex, or sometimes an articulation point.
- Identification of such vertices can provide a sense of where a network is vulnerable (e.g., in the sense of an attack, where disconnecting produces undesired consequences, such as a power outage in an energy network).
- In the giant component of the yeast network, almost 15% of the vertices are cut vertices.

```
yeast.cut.vertices = articulation.points(yeast.gc)
length(yeast.cut.vertices)
```

```
## [1] 350
```

- A nontrivial graph $G$ is $k$-vertex (k-edge) connected if and only if all pairs of distinct vertices $u, v \in V$ can be connected by $k$ vertex-disjoint (edge-disjoint) paths.
- This result relates the robustness of a graph in the face of removal of its vertices (edges) to the richness of distinct paths running throughout it.
- A graph with low vertex (edge) connectivity therefore can have the paths, and hence any 'information' flowing over those paths, disrupted by removing an appropriate choice of a correspondingly small number of vertices (edges).
- shortest.paths, graph.maxfow, and graph.mincut

# Graph Partitioning

- Partitioning refers to the segmentation of a set of elements into 'natural' subsets.
- More formally, a partition $C = \{C_1, \cdots, C_K\}$ of a finite set $S$ is a decomposition of $S$ into $K$ disjoint, nonempty subsets $C_k$ such that $\cup_{k=1}^{K} C_k = S$.
- In the analysis of network graphs, partitioning is a useful tool for finding, in an unsupervised fashion, subsets of vertices that demonstrate a 'cohesiveness' with respect to the underlying relational patterns.
- A 'cohesive' subset of vertices generally is taken to refer to a subset of vertices that
    1. are well connected among themselves, and at the same time
    2. are relatively well separated from the remaining vertices.

- Graph partitioning algorithms typically seek a partition $C = \{C_1, \cdots, C_K\}$ of the vertex set $V$ of a graph $G = (V, E)$ in such a manner that the sets $E(C_k, C'_k)$ of edges connecting vertices in $C_k$ to vertices in $C'_k$ are relatively small in size compared to the sets $E(C_k) = E(C_k, C_k)$ of edges connecting vertices within the $C_k$.
- This problem of graph partitioning is also commonly referred to as community detection in the complex networks literature.
- The use of two well - established classes of methods — those based on adaptations of hierarchical clustering and those based on spectral partitioning.

## Hierarchical Clustering

- A great many methods for graph partitioning are essentially variations on the more general concept of hierarchical clustering
- There are numerous techniques that have been proposed for the general clustering problem, differing primarily in
  1. how they evaluate the quality of proposed clusterings and
  2. the algorithms by which they seek to optimize that quality. These methods take a greedy approach to searching the space of all possible partitions $C$, by iteratively modifying successive candidate partitions.
- Hierarchical methods are classified as either
  - agglomerative, being based on the successive coarsening of partitions through the process of merging, or
  - divisive, being based on the successive refinement of partitions through the process of splitting.
- At each stage, the current candidate partition is modified in a way that minimizes a specified measure of cost.
  - In agglomerative methods, the least costly merge of two previously existing partition elements is executed, whereas
  - in divisive methods, it is the least costly split of a single existing partition element into two that is executed.
- The measure of cost incorporated into a hierarchical clustering method used in graph partitioning should reflect our sense of what defines a 'cohesive' subset of vertices.
  - There are many cost measures that have been proposed.
  - A particularly popular measure is that of modularity.
    - Let $C = \{C_1, \cdots, C_K\}$ be a given candidate partition and define $f_{ij} = f_{ij}(\mathcal{C})$ to be the fraction of edges in the original network that connect vertices in $C_i$ with vertices in $C_j$.
    - The modularity of $\mathcal{C}$ is the value

$$\text{mod}(\mathcal{C}) = \sum_{k=1}^{K} \left[ f_{kk}(\mathcal{C}) - f_{kk}^{*} \right]^2$$

where $f^*_{kk}$ is the expected value of $f_{kk}$ under some model of random edge assignment.
    - $f^*_{kk}$ is defined to be $f_{k+}f_{+k}$, where $f_{k+}$ and $f_{+k}$ are the $k$-th row and column sums of $\mathbf{f}$, the $K \times K$ matrix formed by the entries $f_{ij}$.
    - This choice corresponds to a model in which a graph is constructed to have the same degree distribution as $G$, but with edges otherwise placed at random, without respect to the underlying partition elements dictated by $C$.
  - Large values of the modularity are therefore taken to suggest that $C$ captures nontrivial 'group' structure, beyond that expected to occur under the random assignment of edges.
- In principle the optimization of the modularity requires a search over all possible partitions $C$, which is prohibitively expensive in networks of moderate size and larger.
- A fast, greedy approach to optimization has been proposed, in the form of an agglomerative hierarchical clustering algorithm, and implemented in igraph as fastgreedy.community.
- The result of this and related community detection methods in igraph is to produce an object of the class communities, which can then serve as input to various other functions.

Applying this method to the karate network,

```
kc = fastgreedy.community(karate)
length(kc)
```

```
## [1] 3
```

```
sizes(kc)
```

```
## Community sizes
##  1  2  3
## 18 11  5
```

```
membership(kc)
```

```
##    Mr Hi  Actor 2  Actor 3  Actor 4  Actor 5  Actor 6  Actor 7  Actor 8
##        2        2        2        2        3        3        3        2
##  Actor 9 Actor 10 Actor 11 Actor 12 Actor 13 Actor 14 Actor 15 Actor 16
##        1        1        3        2        2        2        1        1
## Actor 17 Actor 18 Actor 19 Actor 20 Actor 21 Actor 22 Actor 23 Actor 24
##        3        2        1        2        1        2        1        1
## Actor 25 Actor 26 Actor 27 Actor 28 Actor 29 Actor 30 Actor 31 Actor 32
##        1        1        1        1        1        1        1        1
## Actor 33   John A
##        1        1
```

- The largest community of 18 members is centered around the administrator (i.e., John A, vertex ID 34).
- The second largest community of 11 members is centered around the head instructor (i.e., Mr Hi,

vertex ID 1).

```
plot(kc, karate)
```

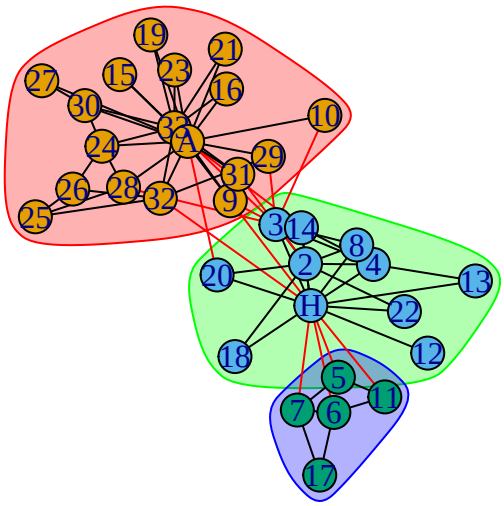


Figure 9: Partitioning of the Karate network obtained from hierarchical clustering

- Whether agglomerative or divisive, when used for network graph partitioning, hierarchical clustering methods actually produce, as the name indicates, an entire hierarchy of nested partitions of the graph, not just a single partition.
- The resulting hierarchy typically is represented in the form of a tree, called a dendrogram.

```
library(ape)
```

```
##
## Attaching package: 'ape'
## The following objects are masked from 'package:igraph':
##
##     edges, mst, ring
```

```
dendPlot(kc, mode="phylo")
```

## Spectral Partitioning

- Another common approach to graph partitioning is to exploit results in spectral graph theory that associate the connectivity of a graph $G$ with the eigen-analysis of certain matrices.
- The graph Laplacian of a graph $G$, with adjacency matrix $A$, is a matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \text{diag}[(d_v)]$ is a diagonal matrix with elements $D_{vv} = d_v$ the entries of the degree sequences of $G$.
- A formal result in spectral graph theory states that a graph $G$ will consist of $K$ connected components if and only if $\lambda_1(L) = \cdots = \lambda_K(L) = 0$ and $\lambda_{K+1}(L) > 0$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N_v}$ are the (not necessarily distinct) eigenvalues of $L$, ordered from small to large.

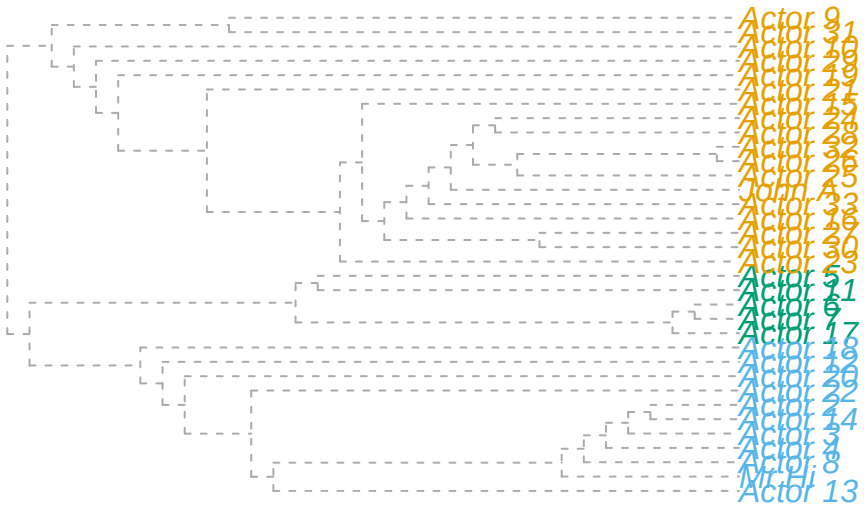


Figure 10: The corresponding dendrogram for this partitioning

- The number of components in a graph is directly related to the number of non-zero eigenvalues of the graph Laplacian.
- The smallest eigenvalue of $L$ can be shown to be identically equal to zero, with corresponding eigenvector $x_1 = (1, \cdots, 1)^T$.
- Therefore, for example, if we suspect a graph $G$ to consist of 'nearly' $K = 2$ components, and hence to be a good candidate for bisection, we might expect $\lambda_2(L)$ to be close to zero.
- In fact, such expectations are reasonable, as the value $\lambda_2$ is closely related to a number of measures of graph connectivity and structure.
- In particular, these relationships indicate that the smaller $\lambda_2$ the more amenable the graph is to being separated into two subgraphs by severing a relatively small number of edges between the two.
- The first to associate $\lambda_2$ with the connectivity of a graph, suggested partitioning vertices by separating them according to the sign of their entries in the corresponding eigenvector $\mathbf{x}_2$.
- The result is to produce two subsets of vertices (a so-called cut)

$$S = \{v \in V : \mathbf{x}_2(v) \geq 0\} \text{ and } \bar{S} = \{v \in V : \mathbf{x}_2(v) < 0\}.$$

  The vector $\mathbf{x}_2$ is hence often called the Fiedler vector, and the corresponding eigenvalue $\lambda_2$, the Fiedler value.

```
k.lap = graph.laplacian(karate)
eig.anal = eigen(k.lap)
```

We plot the eigenvalues of the graph Laplacian

```
plot(eig.anal$values, col="blue", ylab="Eigenvalues of Graph Laplacian")
```

1. there is only one eigenvalue exactly equal to zero (as expected, since this network is connected).

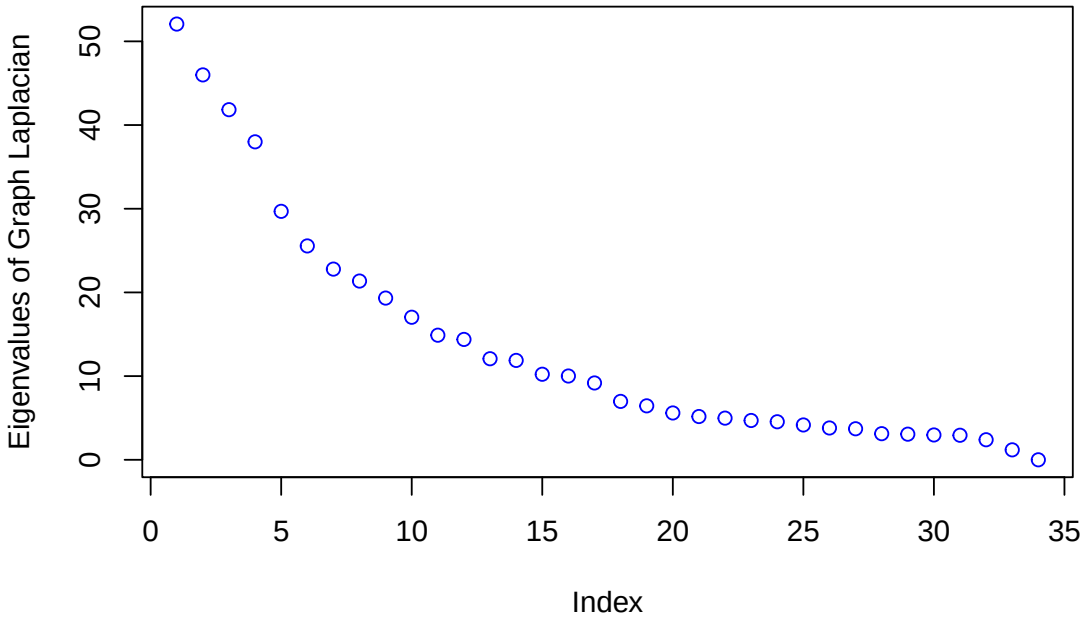


Figure 11: Eigenvalues of Graph Laplacian

2. the second smallest eigenvalue $\lambda_2$ is quite close to zero.

Extracting the Fiedler vector

```
f.vec = eig.anal$vectors[, 33]
```

and plotting the entries of that vector versus actor number,

```
faction = get.vertex.attribute(karate, "Faction")
f.colors = as.character(length(faction))
f.colors[faction == 1] = "red"
f.colors[faction == 2] = "cyan"
plot(f.vec, pch=16, xlab="Actor Number", ylab="Fiedler Vector Entry", col=f.colors)
abline(0, 0, lwd=2, col="lightgray")
```

we find that this spectral method exactly captures the partitioning of the network indicated by the faction labels.

- In general, we may well expect that a network should be partitioned into more than just two subgraphs.
- The method of spectral partitioning just described can be applied iteratively, first partitioning a graph into two subgraphs, then each of those two subgraphs into further subgraphs, and so on.
- Ideally, however, it is desirable that such iterations be aimed at optimizing some common objective function.
- Newman proposes a method whose technical development parallels that of the spectral bisection method quite closely, but with a matrix related to modularity playing the role of the Laplacian **L**. (leading.eigenvector.community)

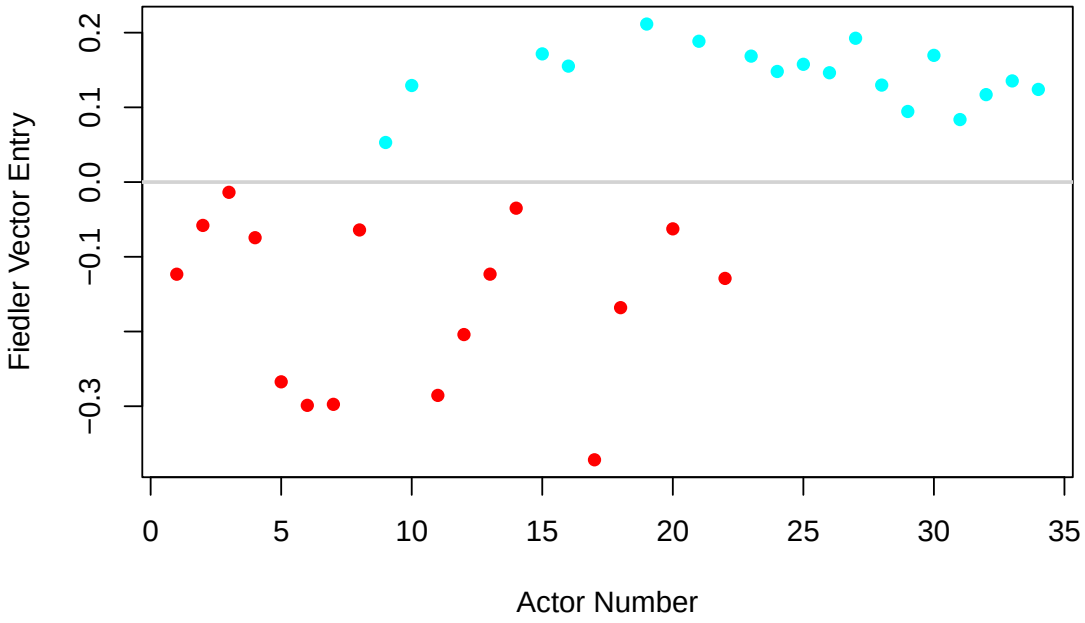


Figure 12: Fiedler vector and its corresponding partition

## Validation of Graph Partitioning

- The question of validation is important for graph partitioning - but often nontrivial.
- It is generally expected, where cohesive subsets of vertices are present in a network graph, that underlying these subsets there is some commonality in certain relevant characteristics (or attributes) of the vertices.
- Graph partitioning may be viewed as a tool for discovering such subsets in the absence of knowledge of these characteristics.
- When we do have knowledge of some externally defined notion of class membership, it can be interesting to compare and contrast the resulting assignments with those deriving from graph partitioning.

```
func.class = get.vertex.attribute(yeast.gc, "Class")
table(func.class)
```

```
## func.class
##   A   B   C   D   E   F   G   M   O   P   R   T   U
##  51  98 122 238  95 171  96 278 171 248  45 240 483
```

- These classes are a way of categorizing the roles of proteins in helping the cell accomplish various tasks through higher-level cellular processes.
- The affinity of proteins to physically bind to each other is known to be directly related to their participation in common cellular functions.
- The external assignment of proteins to functional classes should correlate, to at least some extent, with their assignment to 'communities' by a reasonable graph partitioning algorithm.

```
yc = fastgreedy.community(yeast.gc)
c.m = membership(yc)
table(c.m, func.class, useNA=c("no"))
```

```
##     func.class
## c.m   A  B  C  D  E  F  G  M  O  P  R  T  U
##   1   0  0  0  1  3  7  0  6  3 110  2 35 14
##   2   0  2  2  7  1  1  1  4 39  5  0  4 27
##   3   1  9  7 18  4  8  4 20 10 23  8 74 64
##   4  25 11 10 22 72 84 81 168 14 75 16 27 121
##   5   1  7  5 14  0  4  0  2  3  6  1 34 68
##   6   1 24  1  4  1  4  0  7  0  1  0 19 16
##   7   6 18  6 76  7  9  3  7  8  5  1  7 33
##   8   8 12 67 59  1 34  0 19 60 10  7  6 73
##   9   4  1  7  7  2 10  5  3  2  0  3  0 11
##   10  0  0  0  6  0  0  0  2  0  5  0 11  1
##   11  0  9  0 10  1  3  0  0  0  0  0  2  4
##   12  0  1  3  0  0  0  0  6 10  0  0  0  2
##   13  0  1  1  2  0  1  0  0  2  0  0 16 10
##   14  1  0  4  1  0  1  0  0  4  0  1  0 11
##   15  0  1  0  0  0  2  0  2  0  0  1  0  8
##   16  0  1  2  0  0  1  0  0 10  0  0  0  0
##   17  0  0  1  3  0  0  0  2  0  0  0  2  3
##   18  0  0  0  0  3  1  0  9  0  0  1  0  1
##   19  0  1  1  1  0  0  0  0  0  0  0  0  3
##   20  0  0  0  6  0  0  0  1  0  0  0  1  2
##   21  1  0  0  0  0  0  0  0  6  0  0  1  0
##   22  0  0  0  0  0  0  0  1  0  0  0  0  8
##   23  0  0  0  0  0  0  0  4  0  0  0  0  0
##   24  0  0  0  0  0  0  2  2  0  0  0  1  0
##   25  0  0  0  0  0  0  0  5  0  0  0  0  0
##   26  0  0  1  0  0  0  0  4  0  0  1  0  1
##   27  3  0  4  0  0  1  0  0  0  0  0  0  0
##   28  0  0  0  0  0  0  0  0  0  6  0  0  0
##   29  0  0  0  1  0  0  0  1  0  0  3  0  0
##   30  0  0  0  0  0  0  0  0  0  2  0  0  2
##   31  0  0  0  0  0  0  0  3  0  0  0  0  0
```

## Assortativity and Mixing

- Assortative mixing: Selective linking among vertices, according to a certain characteristic(s)
- Assortativity coefficients: Measures that quantify the extent of assortative mixing. They are essentially variations on the concept of correlation coefficients.
  - The vertex characteristics involved can be categorical, ordinal, or continuous.
  - Consider the categorical case, and suppose that each vertex in a graph $G$ can be labeled according to one of $M$ categories.
  - The assortativity coefficient in this setting is defined to be

$$r_a = \frac{\sum_i f_{ii} - \sum_i f_{i+} f_{+i}}{1 - \sum_i f_{i+} f_{+i}}$$

    where $f_{ij}$ is the fraction of edges in $G$ that join a vertex in the $i$-th category with a vertex in the jth category, and $f_{i+}$ and $f_{+i}$ denote the ith marginal row and column sums, respectively, of the resulting matrix $\mathbf{f}$.
- Assortative Coefficients, $r_a$.
  - $r_a$ lies between $-1$ and 1.
  - It is equal to zero when the mixing in the graph is no different from that obtained through a random assignment of edges that preserves the marginal degree distribution.
  - It is equal to one when there is perfect assortative mixing (i.e., when edges only connect vertices of the same category).
  - However, in the event that the mixing is perfectly disassortative, in the sense that every edge in the graph connects vertices of two different categories, the coefficient need not take the value $-1$.
- The fact that physical binding of proteins is known to be directly relevant to functional classes suggests that there will frequently be strong assortative mixing in protein-protein interaction networks with respective to these classes as attributes.

```
assortativity.nominal(yeast, (V(yeast)$Class=="P")+1, directed=FALSE)
```

```
## [1] 0.4965229
```

- When the vertex characteristic of interest is continuous, rather than discrete, denote by $(x_e, y_e)$ the values of that characteristic for the vertices joined by an edge $e \in E$.
- A natural candidate for quantifying the assortativity in this characteristic is just the Pearson correlation coefficient of the pairs $(x_e, y_e)$,

$$r = \frac{\sum_{x,y} xy(f_{xy} - f_{x+} f_{+y})}{\sigma_x \sigma_y}$$

```
assortativity.degree(yeast)
```

```
## [1] 0.4610798
```