ST5225: Statistical Analysis of Networks
Lecture 4: Descriptive Statistics

WANG Wanjie
staww@nus.edu.sg

Department of Statistics and Applied Probability
National University of Singapore (NUS)

Thursday 8th February, 2018

Outline

- Review
- Eigenvector Centrality
- Cohesion
- Graph Partitions

Review

- Degree sequence/distribution
  - Diameter

- Centrality
  - Closeness

$$c_A = \frac{1}{\text{average distance from node } A \text{ to other nodes}}$$

  - Betweenness

$$c_A = \sum_{u,v \in V \times V} \frac{\sigma(u,v|A)}{\sigma(u,v)}$$

Eigenvector Centrality

- If you know more people who are "centers" of the network, then you are more likely to be a "center"
- Define the "importance" of a node according to its neighbors. Hope it satisfies

$$Cv(i) = \sum_{j \in \{\text{neighbors of } i\}} v(j)$$

- Note: for undirected graphs, C cannot be 1.
  If $C = 1$, then $v(i) = \sum_{j \in \{\text{neighbors of } i\}} v(j)$ for all i. Say A and B are neighbors, then

$$v(A) = V(B) + \sum_{j \in \{\text{neighbors of } A\}, j \neq B} v(j) > v(B),$$

  and

$$v(B) = V(A) + \sum_{j \in \{\text{neighbors of } B\}, j \neq A} v(j) > v(A).$$

  Contradiction!

Eigenvector Centrality

- Recall the adjacency matrix A, where $A_{ij} = 1$ if there is $j \in \{\text{neighbors of } i\}$, and $A_{ij} = 0$ if not. So we rewrite the formula as

$$\begin{aligned} Cv(i) &= \sum_{j \in \{\text{neighbors of } i\}} A_{ij} v(j) + \sum_{j \notin \{\text{neighbors of } i\}} A_{ij} v(j) \\ &= \sum_j A_{ij} v(j). \end{aligned}$$

- Rewrite the formula as

$$v = \alpha A v.$$

  Obviously, this indicates an eigenvector of A.

Eigenvector Centrality

When A has non-negative entries:

- The largest eigenvalue is positive
- The eigenvector corresponding to the largest eigenvalue is non-negative
- There is one such eigenvector for each connected component.

- The eigenvector is called the *eigenvector centrality* for the nodes.
- Note: the eigenvector is $|V| \times 1$ vector, so for node i, the i-th element of the eigenvector is the *eigenvector centrality* of node i.
- Variates of it is largely used in search engines (google...)

Eigenvector Centrality

When A has non-negative entries:

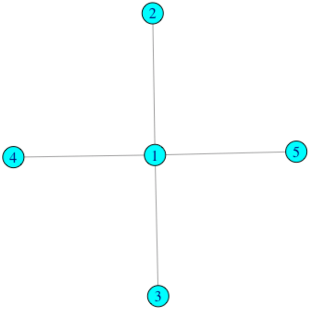
- The largest eigenvalue is positive
- The eigenvector corresponding to the largest eigenvalue is non-negative
- There is one such eigenvector for each connected component.

- The eigenvector is called the *eigenvector centrality* for the nodes.
- Note: the eigenvector is $|V| \times 1$ vector, so for node i, the i-th element of the eigenvector is the *eigenvector centrality* of node i.
- Variates of it is largely used in search engines (google...)

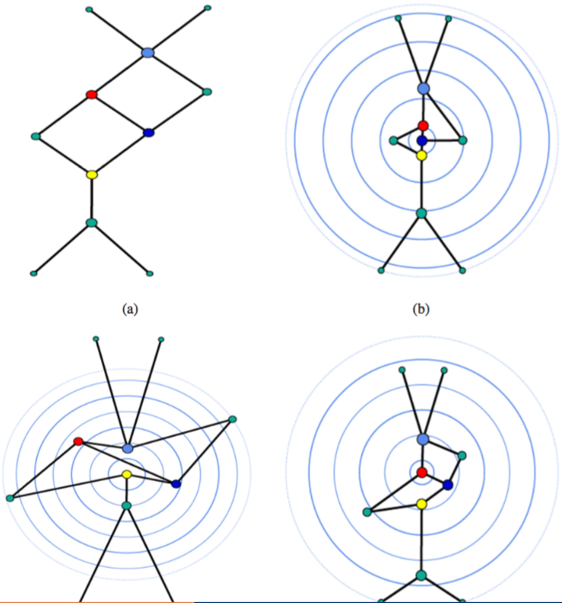
The adjacency matrix for this graph is

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The eigenvector is $(1, 0.5, 0.5, 0.5, 0.5)^T$



3 Types of Centrality



- For directed graphs,
  - the betweenness centrality and closeness centrality can be defined in the same way
  - the eigenvector centrality can be adjusted in two ways: find the eigenvector of $M_{hub} = AA^T$, or of $M_{auth} = A^T A$. It is called the "Hubs and Authorities" algorithms.
- All the notions can be generalized to the *edge centrality*.
  - Generate the dual graph of G, say $G' = (V', E')$, where each node in V' is an edge in G, and each edge in E' is a node in G
  - The edge centrality for E is defined as the vertex centrality for V', correspondingly.

Network Cohesion

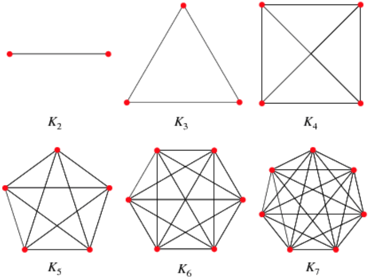
- Interested in *a subset of nodes*: whether these nodes are cohesive
- Interested in the whole network: *How to evaluate the cohesive parts in this network?*
- Examples:
  - If both B and C are friends of A, are B and C friends?
  - Does the structure of the internet pages tend to separate, with respect to distinct types of content?

- We need some measures for the network cohesion. Again, to describe it, there are multiple measurements, including
  - Densities (cliques, cores, local density)
  - Hierarchical structure (clusters)

Cliques

Recall:

- Cliques: A subgraph which is complete
  - Cliques are *fully cohesive*. Distance between each two nodes is 1.
- Examples of Cliques:

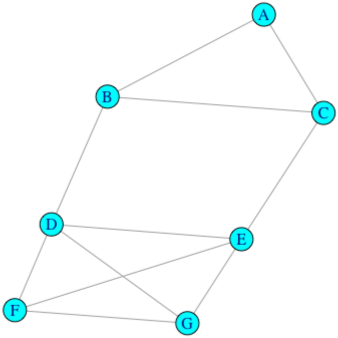


- If $H \subset G$ is a clique, then any induced subgraph of H is a clique.
- Maximal clique: a clique that no larger clique contains it

Cliques

What are the maximal cliques in the following network?
Solution: Find the maximal clique for each node

- For A, the maximal clique containing A is the subgraph formed by $\{A, B, C\}$
- For B, there are two maximal cliques, the one formed by $\{A, B, C\}$ and the one formed by $\{B, D\}$
- For C, there are also two maximal cliques, $\{A, B, C\}$ and $\{C, E\}$
- For D, the one formed by $\{B, D\}$ and the one formed by $\{D, E, F, G\}$
- For E, similarly we have $\{C, E\}$ and $\{D, E, F, G\}$
- For F and G, the maximal clique is only the one formed by $\{D, E, F, G\}$.



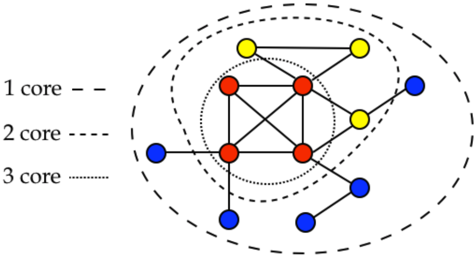
- The nodes in the same clique can be seen as a small "community". They connect with each other in this community.
- If a network has a large clique, then this network is more "cohesive"
- If there are a lot of edges ($|E| > (|V|^2/2)\frac{n-2}{n-1}$), there must be a clique (with size n).
- Computation cost:
  - Given a clique, whether it is maximal or not can be done in $O(|V| + |E|)$ time
  - Whether a graph has a maximal clique of at least size n is NP-complete (nondeterministic polynomial time)
  - NP-complete: e.g. the computation cost is $2^{|V|}$. For large networks where $|V|$ and $|E|$ are large, it is impossible to realize

- The clique requires connection between every pair of nodes – very strict
- Relaxation: every node has a high degree.

k-core

A subset of nodes is called a *k-core*, if in the induced subgraph, all nodes have degree at least k.

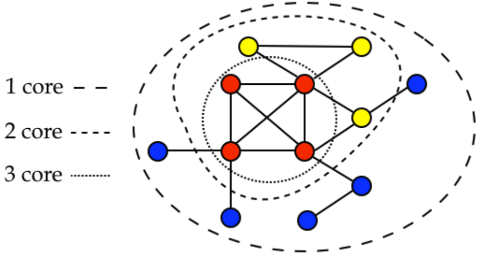
- Example[1]:



[1] https://chaoslikehome.wordpress.com/tag/k-core/

Maximal k-core

The *(maximal) k-core* is a k-core that cannot be enlarged to a larger k-core.



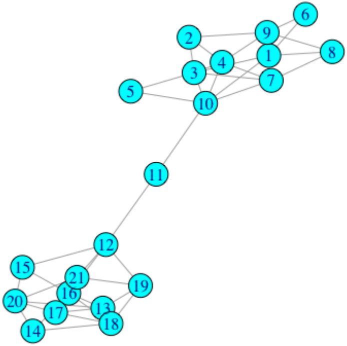
- If a node is in the k-core, it is also in the $k-1$-core.
- We can also define the *coreness of a vertex i*:

$$\text{Coreness}(i) = \max\{k : i \in \ k\text{-core }\}$$

- k-core can be found by deleting the node with smallest degrees

Recall:

- The network can be decomposed into several connected components
- The nodes in each component are connected. Nodes in different components are disconnected.
- It is possible that if several nodes/edges are deleted, the originally connected components are separated



Vertex-Connectivity

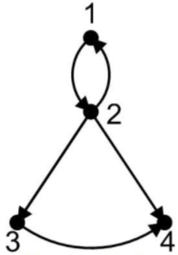
A graph G is called *k-vertex-connected* if

1. the number of nodes $|V| > k$;
2. the removal of any subset of vertices $X \subset V$ of cardinality $|X| < k$ leaves a subgraph $G - X$ that is connected.

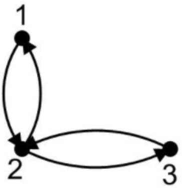
- The smallest number of nodes you have to remove to disconnect the graph
- The previous graph is 1-vertex-connected
- $k < \min_i d(i)$, otherwise you can isolate the node with smallest degree
- Similarly, we can define *k-edge-connected* graphs.
  1. the number of edges $|E| > k$;
  2. the removal of any subset of edges $X \subset E$ of cardinality $|X| < k$ leaves a subgraph $G - X$ that is connected.

Connectivity

- The connectivity can be expanded to the directed graphs straightforwardly.
  - *Weakly connected.* If the underlying graph (the undirected graph where the labels 'tail' and 'head' are removed from G) is connected, G is called *weakly connected.*
  - *Strongly connected.* If every node v is reachable from every other node u by a *directed* walk, G is called *strongly connected.*
- Example[2]



Weakly connected graph



Strongly connected graph

- Vertex/Edge-connectivity can be extended analogously.

[2] http://slideplayer.com/slide/2433835/

- For the directed graph, there is a new characterization, a 'bowtie'.

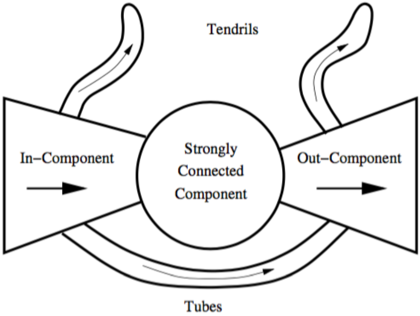


Fig. 4.5 'Bowtie' structure of a directed network graph. Adapted from Broder et al. [67].

- Strongly connected component (SCC)
- *in-component*, nodes can reach SCC, but cannot be reached from SCC
- *out-component*, nodes can be reached from the SCC but cannot reach SCC
- *tubes*, nodes between the in- and out- components, not SCC
- *tendrils*, nodes that can neither be reach nor be reached from the SCC

Connectivity

- The in- and out-components are in some sense 'upstream' and 'downstream' from the SCC
- First discovered when studying the WWW graph
  - Crawling on webpages, and record the hyperlinks in the pages
  - The hyperlinks are record as edges, while the webpages are the nodes
  - Several papers found this nature, and also studied on the reason
- Also found in other real data sets

Local Density

- For an undirected graph $G = (V, E)$, the possible number of edges is $\binom{|V|}{2} = \frac{|V|(|V|-1)}{2}$
- The density is the proportion of the truly observed edges

Local Density

For a graph G, the *density* of G is

$$den(G) = \frac{|E|}{|V|(|V|-1)/2}.$$

- Prob(two randomly picked nodes are connected by an edge)
- Recall the average degree $\bar{d}(G) = \frac{2|E|}{|V|}$, the density is just a rescaling of $\bar{d}(G)$ of G, where

$$den(G) = \frac{|E|}{|V|(|V|-1)/2} = \frac{\bar{d}(G)}{|V|-1}$$

Local Density

- The definition can also be applied to an induced subgraph $H = (V_H, E_H) \subset G$, where

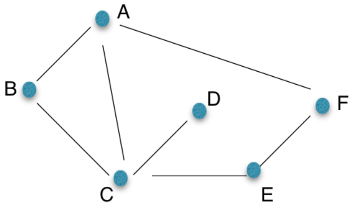
$$den(H) = \frac{|E_H|}{|V_H|(|V_H| - 1)/2}.$$

It can be seen as the local density at subgraph H.

- Especially, for node v, take the subgraph $H = H_v$, which contains the nodes $\{v$ and neighbors of $v\}$ and the edges between them.
- Define the *density of node v* as

$$den(v) = den(H_v)$$

Example.



- Density of the whole graph: $\frac{7}{6*5/2} = 7/15 = 0.47$
- $Den(H_A) = \frac{4}{4*3/2} = 2/3$; $Den(H_B) = \frac{3}{3*2/2} = 1$;
  $Den(H_C) = \frac{5}{5*4/2} = 0.5$; $Den(H_F) = \frac{2}{3*2/2} = 2/3$.

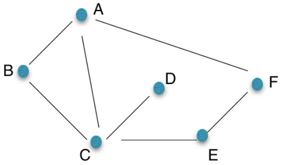
- Obviously, $0 \leq den(G) \leq 1$, as it is a proportion
- $Den(G) = 1 \iff G$ is a clique

- Call the 3-node complete graph as a *triangle*
- Call a 2-star graph as *connected triple*
- Note: The connected triple is one edge less than the triangle

- For a node v with $d(v) \geq 2$, define

$$cl(v) = \frac{\#\text{triangles } v \text{ falls into}}{\#\text{connected triples that both edges are incident to } v}$$

- Example.



$$cl(A) = 1/3, \quad cl(B) = 1, \quad cl(C) = 1/6, \quad cl(E) = 0, \quad cl(F) = 0$$

Local Density

- If a group contains more triangles, the group is more "dense"
- For a graph $G = (V, E)$, consider $V' \subset V$, which contains all the nodes with degrees ≥ 2. Each of these nodes have a
- Define the *clustering coefficient* for the graph as

$$cl(G) = \frac{1}{V'} \sum_{v \in V'} cl(v)$$

- However, this is not quite informative, so a weighted one is more generally used, which is

$$cl(G) = \frac{\sum_{v \in V'} \tau_3(v) cl(v)}{\sum_{v \in V'} \tau_3(v)} = \frac{3\tau_{\triangle}(G)}{\tau_3(G)},$$

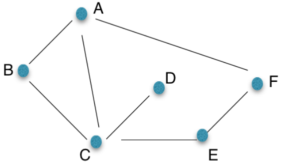
where $\tau_3(v) = \#$ connected triples that two edges are incident with v; $\tau_3(G) = \#$ connected triples in G; and $\tau_{\triangle}(G) = \#$ triangles in G.

Local Density

- Show the equality:

$$
\begin{aligned}
cl(G) &= \frac{\sum_{v \in V'} \tau_3(v) cl(v)}{\sum_{v \in V'} \tau_3(v)} \\
&= \frac{\sum_{v \in V'} \tau_3(v) \times \tau_\triangle(v)/\tau_3(v)}{\sum_{v \in V'} \tau_3(v)} \\
&= \frac{\sum_{v \in V'} \tau_\triangle(v)}{\tau_3(G)} \\
&= \frac{3\tau_\triangle(G)}{\tau_3(G)}
\end{aligned}
$$

Example



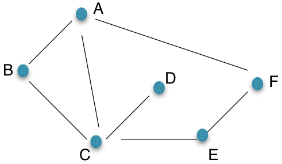
$$\tau_3(G) = 12, \quad \tau_\triangle(G) = 1, \quad cl(G) = 0.25$$

Local Density

- Show the equality:

$$
\begin{aligned}
cl(G) &= \frac{\sum_{v\in V'} \tau_3(v) cl(v)}{\sum_{v\in V'} \tau_3(v)} \\
&= \frac{\sum_{v\in V'} \tau_3(v) \times \tau_\triangle(v)/\tau_3(v)}{\sum_{v\in V'} \tau_3(v)} \\
&= \frac{\sum_{v\in V'} \tau_\triangle(v)}{\tau_3(G)} \\
&= \frac{3\tau_\triangle(G)}{\tau_3(G)}
\end{aligned}
$$

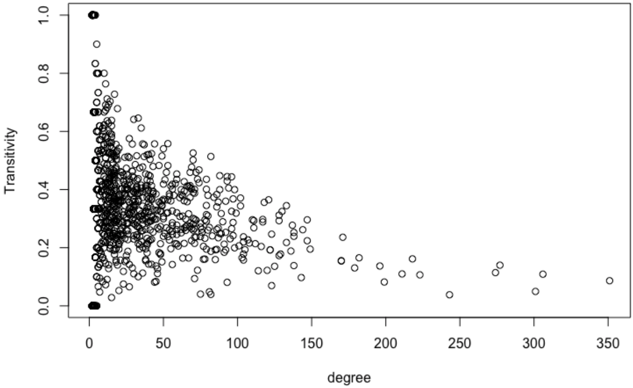
Example



$$\tau_3(G) = 12, \quad \tau_\triangle(G) = 1, \qquad cl(G) = 0.25$$

- Clustering coefficient is also called *transitivity*. For one node v, it means if v knows u and v knows w, what is the chance that u knows w.
- For the whole network, it is the conditional prob. of three nodes knowing each other, given that one knows the two others
- Transitivity for some typical graphs:
  - Transitivity for k-stars are always 0.
  - Transitivity for k-rings are always 0.
  - Transitivity for a complete graph is always 1.
- For large-scale real networks, most of the times (of course not all!)
  - $cl(v)$ varies *inversely* with vertex degree

Recall: Political Blogs, 1490 nodes, 16715 edges
The Degree-Clustering Coefficient figure is as following



For the whole network, the transitivity is 0.226.

- The graph is not uniformly dense, usually it contains several more cohesive parts

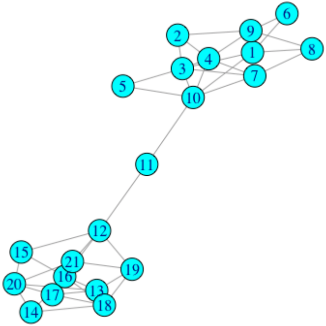
Partition

For a set V, a *partition* of V is a set of subsets $\mathfrak{C} = \{C_1, C_2, \cdots, C_K\}$, where $C_1, C_2, \cdots, C_K$ are disjoint, and $\cup_{i=1}^K C_i = V$.

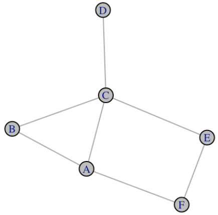
- **Goal**: Find a partition of the graph G, so that in each subset C_i, the nodes are *well* connected, and between the subsets, the nodes are *separated*
- Application:
  - Community detection in social networks
  - Identification of possible protein complexes from protein interaction networks
- Motivation for the stochastic block model (introduce in Lecture 9-12)

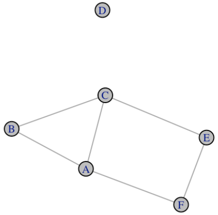
Hierarchical Clustering

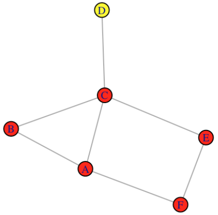
- Betweenness: How many shortest edges will pass this edge
- If there are two communities, the edges between them should have high betweenness



- If we keep on moving the edges with highest betweenness, we will have more components
- Repeat until we achieve the result we need







Remark.

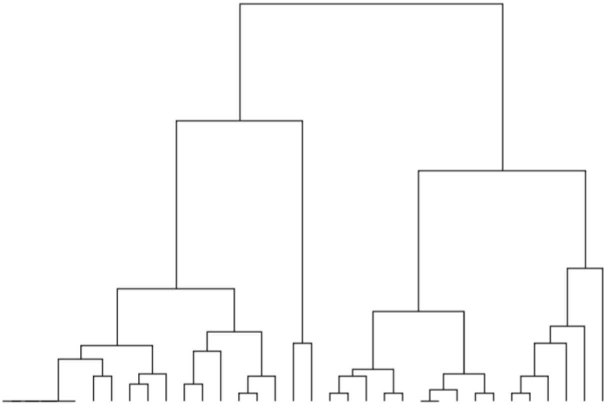
- The process can be repeated until all the nodes are separate (or the result you need according to prior information, say, we already know there are 3 communities)
- However, it is easy to cause the isolation of some nodes.

Questions and Answers:

- How about other measurements, instead of betweenness?
  - Need a *cost function* to *evaluate the quality* of the clustering/partitions
  - Various of cost functions, giving different results
- Dividing nodes, or merging nodes?
  - Both ways work
  - Starting with the original graph, and dividing nodes. We call this as a *divisive* method.
  - Starting with graph $|V|$ clusters (each node is a cluster), and merging nodes. We call this a *agglomerative* method.
- Is this the optimal result?
  - Not necessarily. The optimal result comes from exhaustive search, which is time consuming. This is a greedy search method, which may stuck at local maxima.

Questions and Answers:

- Record the result after each step
  - It will create the entire hierarchy of nested partitions, in the form of a tree. We call it a *dendrogram*.



3

[3]SAND, Fig 4.7 Hierarchical clustering of the karate club network.

Most cost functions is a measure of *(dis)similarity* between sets, based on node-node (dissimilarity).

- Node-node (dis)similarity examples:
  - Euclidean distance:

$$dist(v_i, v_j) = \sqrt{\sum_{k \neq i,j} (A_{ik} - A_{jk})^2}, \quad A_{ij} : (i,j)th \text{ element of adjacency matrix } A$$

    When v_i and v_j have larger distance, the neighbors of them are less shared, and so v_i and v_j are possibly to be separated.
  - Another dissimilarity measurement:

$$dis(v_i, v_j) = \frac{\#\text{unshared neighbors by } v_i \text{ and } v_j}{\text{largest degree} + \text{second largest degree}}$$

    - Numerator: number of nodes that are either neighbors of v_i only, or neighbors of v_j only
    - Ranges in $(0, 1)$

Cost Functions

Begin with the node-node (dis)similarity, different ways to calculate the (dis)similarity for a partition $\mathfrak{C}$.

- Single Linkage

  For partition $\mathfrak{C} = \{C_1, C_2, \cdots, C_K\}$, the dissimilarity between two elements C_1 and C_2 is defined as

  $$dis(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2} dis(v_i, v_j),$$

  the *minimal* distance between two nodes in the two subsets.

- Complete Linkage

  The dissimilarity between two subsets C_1 and C_2 is defined as

  $$dis(C_1, C_2) = \max_{v_i \in C_1, v_j \in C_2} dis(v_i, v_j),$$

  the *maximal* distance between two nodes in the two subsets.

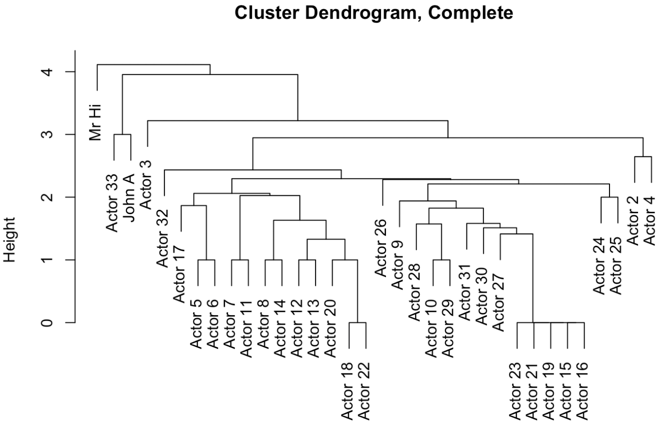
- Average Linkage

  The dissimilarity between two subsets C_1 and C_2 is defined as

  $$dis(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{v_i \in C_1, v_j \in C_2} dis(v_i, v_j),$$

  the *average* distance between all pairs of nodes in the two subsets.

HC with Complete Linkage: Karate Club



Note: cut off at the number of communities you want

Cost Functions

The cost can be defined on the partition directly, not through the node-node (dis)similarity.
For example,

- *modularity* of a partition $\mathfrak{C} = \{C_1, C_2, \cdots, C_K\}$ of the node set V.
- Define the function $f_{ij}(\mathfrak{C})$, where

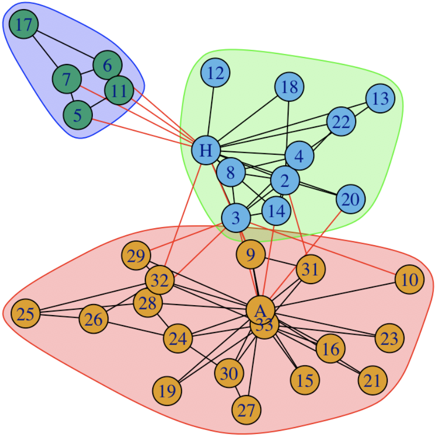
$$f_{ij}(\mathfrak{C}) = \frac{|\{(v_k, v_l) \in E; v_k \in C_i, v_l \in C_j\}|}{|E|},$$

  the fraction of edges that connect a node in C_i and a node in C_j.
- The modularity is

$$mod(\mathfrak{C}) = \sum_{i=1}^{K} [f_{ii}(\mathfrak{C}) - f_{ii}^*], \qquad \text{where } f_{ii}^* = (\sum_{j \in V} f_{ij})(\sum_{j \in V} f_{ji}),$$

- Note:
  - For undirected graphs, $f_{ij} = f_{ji}$
- Problem: non-polynomial computation time

HC with Modularity: Karate Club



Note: 3 communities

Summary: Graph partitions

- A phenomenon in many applications: biology, social network, etc.
- Hierarchical clustering: not sure where to cut off; may stuck at the local minima
- Modularity: computation cost
- Still many other methods:
  - minimal spanning tree
  - k-means
  - Spectral clustering
  - etc.
- The attributes may help