

While most network analysis revolves around **vertex-based** characteristics—like degree and centrality—some questions are more naturally associated with **edges**.

Examples include:

- Which relationships are most important for the spread of information?
- Which connections serve as critical bridges within a network?

Edge betweenness centrality

Edge betweenness centrality extends the idea of vertex betweenness to edges. It assigns a value to each edge reflecting the number of shortest paths that traverse it.

R Example (Karate Club Network): *Shortest path.*

```
> eb <- edge.betweenness(karate)
# Identify edges with the top 3 highest values
> E(karate)[order(eb, decreasing = TRUE)[1:3]]
```

Edge sequence:

```
[53] John A -- Actor 20
```

```
[14] Actor 20 -- Mr Hi
```

```
[16] Actor 32 -- Mr Hi
```

> edge가 가장 짧음! ↑.

Interpretation: These results indicate that Actor 20 plays a key role in facilitating the flow of information between the head instructor (Mr Hi, vertex 1) and the administrator (John A, vertex 34). The edges involving Actor 20 lie on many shortest paths between different parts of the network.

Beyond Betweenness: Edge Centrality via Line Graphs

Not all vertex centrality measures translate directly to edges.
However, there's a workaround using the concept of a **line graph**.

- The **line graph** of a network G , denoted $G' = (V', E')$, transforms each edge of G into a vertex in G' .
- Two vertices in G' are connected if their corresponding edges in G are incident to a common vertex.

서로 만나고 있는 행위.

Beyond Betweenness: Edge Centrality via Line Graphs

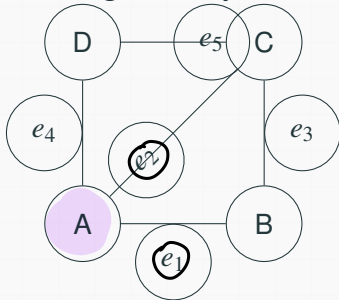
In `igraph`, the line graph can be constructed using:

```
> line.graph(G)
```

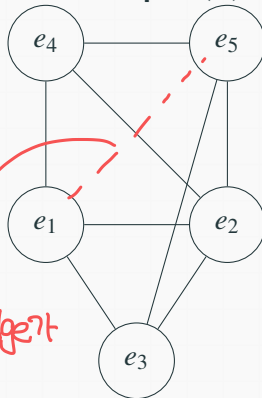
This transformation allows us to apply traditional vertex centrality measures to edges, by analyzing the line graph of the original network.

Line Graph Example

Original Graph G



Line Graph $L(G)$



연결된 edge가
있음.

Characterizing Network Cohesion

Network Cohesion: clique : 얼마나 잘 연결되어있는지!

One foundational way to characterize **network cohesion** is to look for **specific subgraph structures** that indicate tight interconnection.

A classic example is the **clique**, a subset of vertices where every pair is connected — in other words, **a complete subgraph**.

Conducting a census of subgraphs like cliques provides a structural snapshot of how cohesive a network is.

Completed graph의 subgraph 수 \Rightarrow 많이 연결!

Network Cohesion: clique

Using the karate network, a clique census can be performed using:

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

Result:

- 34 cliques of size 1 (individual nodes)
- 78 cliques of size 2 (edges)
- 45 cliques of size 3 (triangles)
- 11 cliques of size 4
- 2 cliques of size 5

	2	3	4	5
⇒	34	78	11	2

Network Cohesion: clique

The two largest cliques (size 5) are:

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

```
[[1]] 1 2 3 4 8 (1,2,3,4) ... 2개의 4개 vertex #.
```

```
[[2]] 1 2 3 4 14 (1,2,3,8)
```

7. (2,3,4,8): maximal clique. : ?

These two cliques share four actors in common, including actor 1 (the head instructor).

The size of the largest clique in a graph is called the **clique number**.

Maximal vs. Maximum Cliques

- A **maximal clique** is not a subset of any larger clique.
- A **maximum clique** is the largest possible clique in the graph.

↳ 그래프가 가질 수 있는 complete clique 중의 크기가 가장 큰 거.

We can get the sizes of all maximal cliques:

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

2 3 4 5
11 21 2 2

↳ ~~count~~ X
⇒ 가장 clique 계산 : count 계산.

Large cliques are rare in real-world networks, since real-world networks are often sparse. For instance,

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

[1] 23

The maximum clique in this much larger network has 23 vertices, despite the overall size being much greater than the karate network.

공기/사향 향이

A **k-core** is a maximal subgraph where every vertex has degree at least k , and such that no other subgraph obeying the same condition contains it (i.e., it is the maximal in its property). It's a relaxed version of a clique and useful for visual decomposition.

K-core decompositions often appear as layers (like onions), and are helpful in visualizing core-periphery structure.

K 이상의 degree를 가진 노드.

```
> cores <- coreness(karate)
> sna::gplot.target(g, cores, circ.lab = FALSE,
+   circ.col = "skyblue", usearrows = FALSE,
+   vertex.col = cores, edge.col = "darkgray")
```

K-Cores

동일한
↓
같은 자음
↻
같은 vertex.

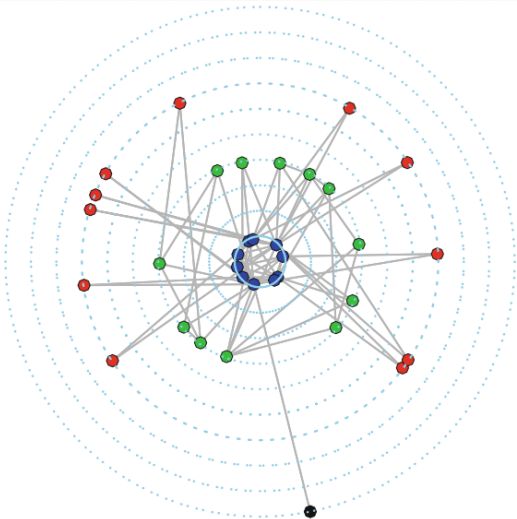


Figure 1: Karate Club K-core Example.

Dyads and Triads (in Directed Graphs)

on & off.

In directed graphs:

- A **dyad** is a pair of vertices, which can be:

- **Null** — no edge between them
- **Asymmetric** — one directed edge
- **Mutual** — both directions present

- A **triad** is a triple of vertices; there are 16 possible configurations 3741.

A census of the possible states of these two classes of subgraphs, i.e., counting how many times each state is observed in a graph G , can yield insight into the nature of the connectivity in the graph.

Dyads and Triads (in Directed Graphs): example

Example from the AIDS blog network:

• triad.

```
> aidsblog <- simplify(aidsblog)
```

```
> dyad_census(aidsblog)
```

```
$mut
```

```
[1] 3
```

```
$asym
```

```
[1] 177
```

```
$null
```

```
[1] 10405
```

10000~의 possible edge: 아주 많음 X: 10405

한쪽 연결 : 177.

쌍방향 : 3.

Connectivity. 강하게 연결이 아니다.

$A \rightarrow B$

$A \leftarrow B$

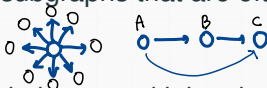
$A \quad B$

$A \leftrightarrow B$

This shows most dyads are either null or asymmetric—a one-sided linking pattern.

Motifs

Motifs are small, recurring subgraphs that are often biologically meaningful.



- **Fan:** A single vertex pointing to multiple others
- **Feedforward loop:** Three vertices with edges $\{u,v\}$, $\{v,w\}$, and $\{u,w\}$

Because motif enumeration is computationally intensive, sampling is often used:

\Rightarrow ~~graph~~ $\times \Rightarrow$ sampling

```
> graph.motifs(..., sample = ...)
```


Density and Related Notions of Relative Frequency

The **density** of a graph measures how many edges exist relative to the number of possible edges.

For an undirected graph $G = (V, E)$ with no self-loops:

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

For directed graphs, the denominator becomes $|V_H|(|V_H| - 1)$.

✓ This quantity ranges from 0 (no edges) to 1 (a clique).

H is a subgraph of G .

Ego-centric Graph

An **ego-centric graph** focuses on a particular vertex (called the **“ego”**) and includes the vertex's immediate neighbors (called the **“alters”**).

In this type of graph: - The ego is the central node.

- The alters are the nodes directly connected to the ego.
- The edges represent relationships between the ego and its neighbors.

Ego-centric graphs are useful in network analysis because they allow us to explore the local structure surrounding a particular node, offering insights into that node's immediate environment and relationships.

Ego-centric Graph

Example: Ego Subgraphs in the Karate Network.

The density of the full network versus two ego-centric subgraphs (around vertices 1 and 34):

```
> 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
```

These ego-networks are noticeably denser than the full network.

Global Clustering (Transitivity)



↪ The **clustering coefficient** (also called **transitivity**) is defined as:

$$0 \leqslant \text{cl}_T(G) = \frac{3 \times \text{Number of triangles}}{\text{Number of connected triples}} \leqslant 1$$

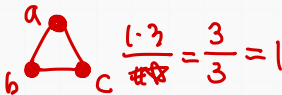
It reflects how often connected triples close into triangles.

> transitivity(karate)

[1] 0.2556818

Number of 2-star graph.
sub

About 25.6% of connected triples close into triangles.



Local Clustering Coefficient

vertex degree:

For a vertex v , let: - $\tau_3(v) = \binom{d_v}{2}$: number of connected triples centered at v - $\tau_\Delta(v)$: number of triangles involving v

Then,

답하기 3
양자나
문제 왜 0~1 ?

$$cl(v) = \frac{\tau_\Delta(v)}{\tau_3(v)}$$

```
> transitivity(karate, "local", vids = c(1,34))  
[1] 0.1500000 0.1102941
```

The instructor and administrator each have lower local clustering than the global network.

Reciprocity (Directed Graphs)

↑ 상반적인가?

Reciprocity measures how often directed edges are reciprocated.

Two definitions:

asym 대비 mutual 정도.

- *Dyadic*: Ratio of mutual dyads to asymmetric dyads.

- *Edge-based*: Ratio of reciprocated edges to total edges. 더 많이 적음.
total 중 얼마나 ~.

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

```
[1] 0.03278689
```

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

```
[1] 0.01666667
```

The AIDS blog network shows low reciprocity.

✓ Weakly Connected

Connectedness and Components (Maximal : 같은 성질에 대해 가장 큰 set)

A graph is **connected** if there is a path between every pair of vertices.

```
> is.connected(yeast)
[1] FALSE
```

Giant component: The largest connected component in a graph.

```
> comps <- decompose.graph(yeast)
> table(sapply(comps, vcount))
 2   3   4   5   6   7 2375
63  13   5   6   1   3   1
```

174 이상의 component

↳ 엄청 큰! + 4개!!

The largest component contains $2375/2617 \approx 90\%$ of nodes.

Small-world property

Exhibits “small-world” property

- Short average path lengths
- High clustering

```
> yeast.gc <- decompose.graph(yeast)[[1]]
```

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

```
[1] 5.09597
```

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

```
[1] 15
```

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

```
[1] 0.4686663
```

Connected graph의 diameter : 전체 vertex pair의 shortest path. distance 중 최댓값.

Connectivity and Vulnerability

small - world.

11.

Vertex connectivity and **edge connectivity** measure how robust a graph is to the removal of vertices or edges.

- A graph G is said to be **k -vertex-connected** if:

1. $|V| > k$
2. The graph remains connected after the removal of any set of fewer than k vertices

- A graph G is said to be **k -edge-connected** if:

1. $|V| \geq 2$
2. The graph remains connected after the removal of any set of fewer than k edges

임의의 k 개 정점 제거 \rightarrow 연결 $\Rightarrow k$ -edge connected.

$k \uparrow \Rightarrow$ vertex 임의의 k 개 연결 : 견고하다.

Connectivity and Vulnerability

Vertex connectivity and **edge connectivity** measure how robust a graph is to the removal of vertices or edges.

- A graph G is said to be **k -vertex-connected** if:

1. $|V| > k$
2. The graph remains connected after the removal of any set of fewer than k vertices

- A graph G is said to be **k -edge-connected** if:

1. $|V| \geq 2$
2. The graph remains connected after the removal of any set of fewer than k edges



max $k \leq |V|$.

a k -vertex connected

Connectivity and Vulnerability

In both cases, the value of k represents the “connectivity” of the graph. A higher k means greater resilience.

It can be shown that:

연결성 증명.

Vertex connectivity \leq Edge connectivity \leq Minimum vertex degree d_{\min}

```
> vertex.connectivity(yeast.gc)
[1] 1
> edge.connectivity(yeast.gc)
[1] 1
```

The yeast network is minimally connected—only one removal can break it.

Cut Vertices and Articulation Points

An **articulation point** (cutvertex) is a vertex in a graph that, when removed along with all its associated edges, results in a graph with more connected components than the original graph.

i.e., An articulation point disconnects the graph when removed.


```
> yeast.cut.vertices <- articulation.points(yeast.gc)
> length(yeast.cut.vertices)
[1] 350
```

About 15% of nodes are articulation points.

Menger's Theorem

Let $G = (V, E)$ be a finite undirected graph and $u, v \in V$, with $u \neq v$.

- **Vertex version**: The minimum number of vertices (excluding u and v) whose removal separates u from v equals the maximum number of internally vertex-disjoint paths from u to v .
- **Edge version**: The minimum number of edges whose removal separates u from v equals the maximum number of edge-disjoint paths from u to v .



This Theorem essentially states that a nontrivial graph G is k -vertex-connected (or k -edge-connected) if and only if all pairs of distinct vertices $u, v \in V$ can be connected by k vertex-disjoint (or edge-disjoint) paths.

This result relates the robustness of a graph in the face of vertex (or edge) removal to the richness of distinct paths running throughout it. A graph with low vertex or edge connectivity can have those paths - and thus any information flowing along them - disrupted by removing a

Menger's Theorem

Menger's Theorem essentially states that a nontrivial graph G is k -vertex-connected (or k -edge-connected) if and only if all pairs of distinct vertices $u, v \in V$ can be connected by k vertex-disjoint (or edge-disjoint) paths.

This result relates the robustness of a graph in the face of vertex (or edge) removal to the richness of distinct paths running throughout it. A graph with low vertex or edge connectivity can have those paths - and thus any information flowing along them - disrupted by removing a correspondingly small number of vertices or edges.

Directed Graphs: Strong vs. Weak Connectivity

- **Weak connectivity:** Ignoring edge directions, the graph is connected.
- **Strong connectivity:** There is a directed path between every pair of nodes.

```
> is.connected(aidsblog, mode = "weak")  
[1] TRUE  
> is.connected(aidsblog, mode = "strong")  
[1] FALSE
```

The AIDS blog network is weakly/but not strongly connected.

```
> aidsblog.scc <- clusters(aidsblog, mode = "strong")  
> table(aidsblog.scc$ccsize)  
  1    4  
142    1
```

There is only one strongly connected component of size 4.