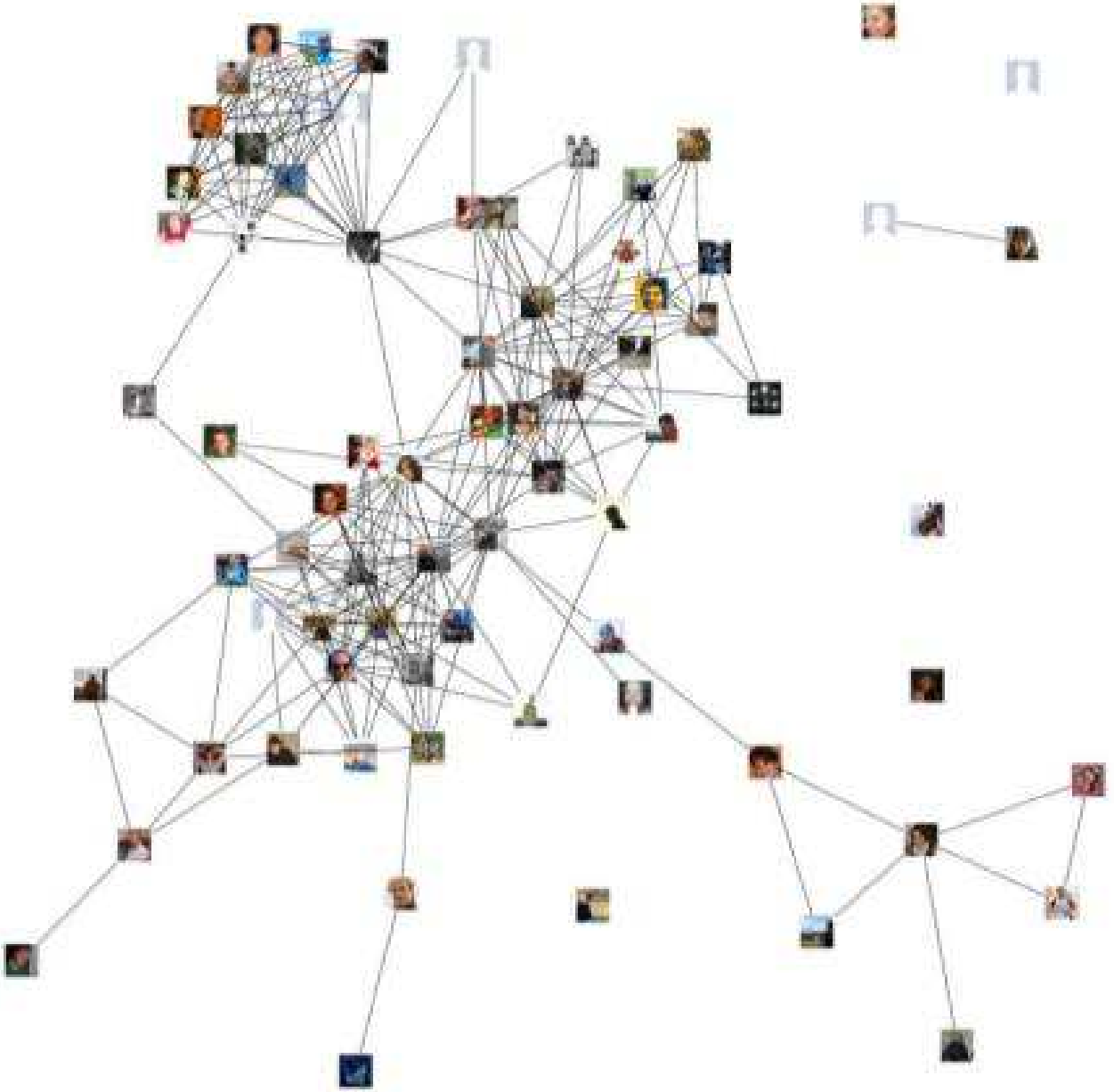
Graph Mining and Community Detection

Jean-Baptiste Durand

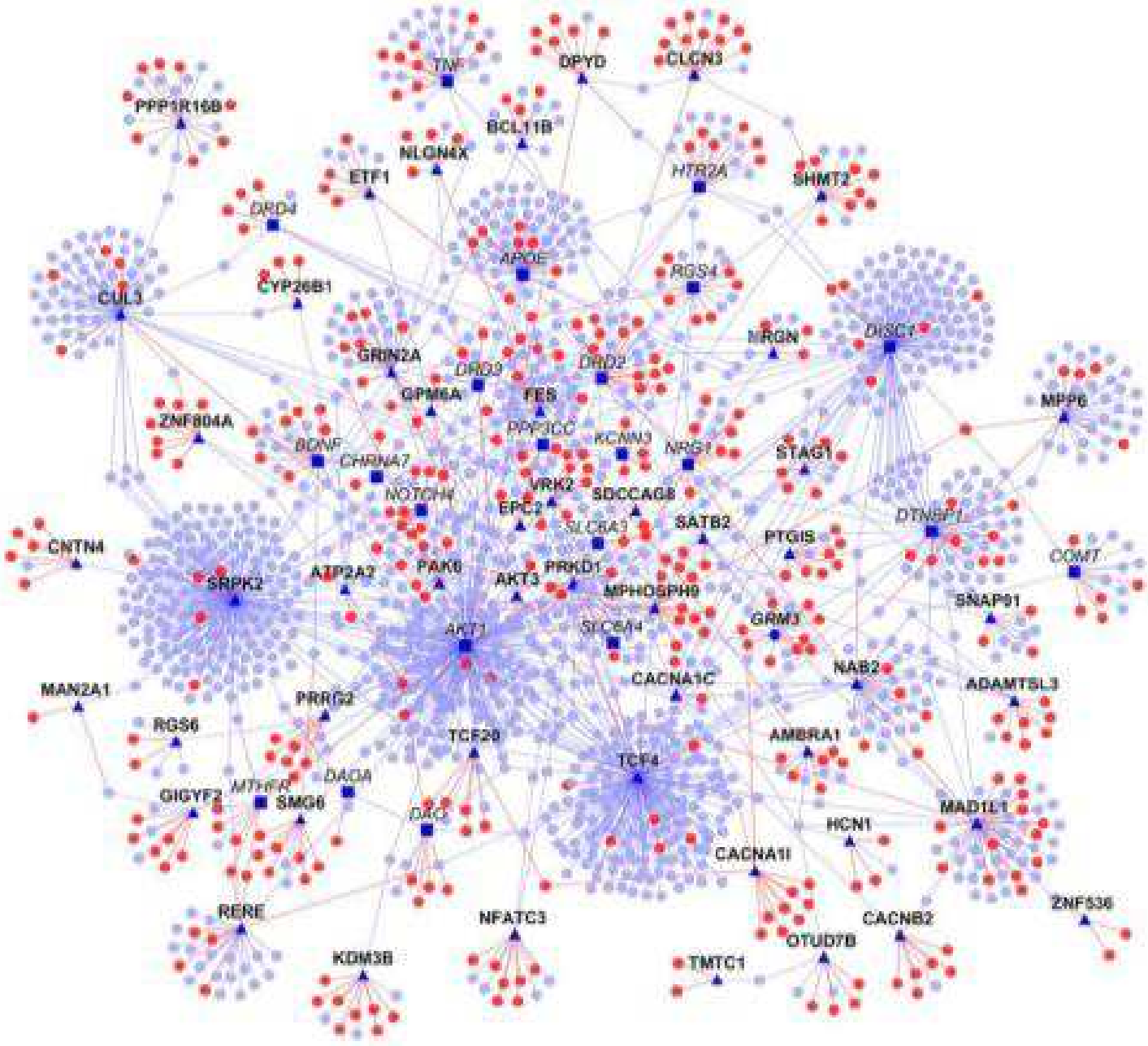
Credits: Charlotte Laclau

Community detection ≡ clustering of graph vertices

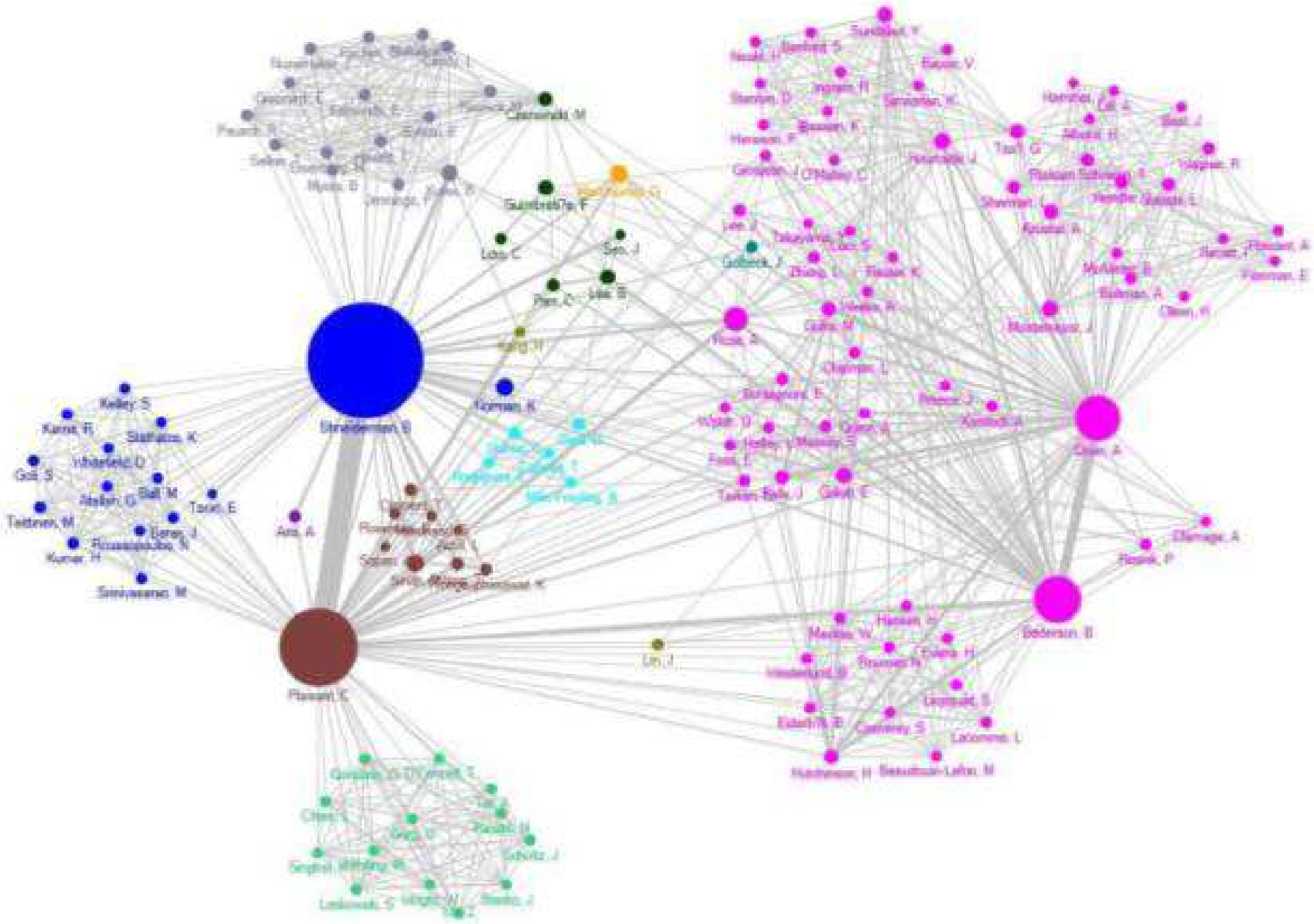
Example 1 : Facebook network



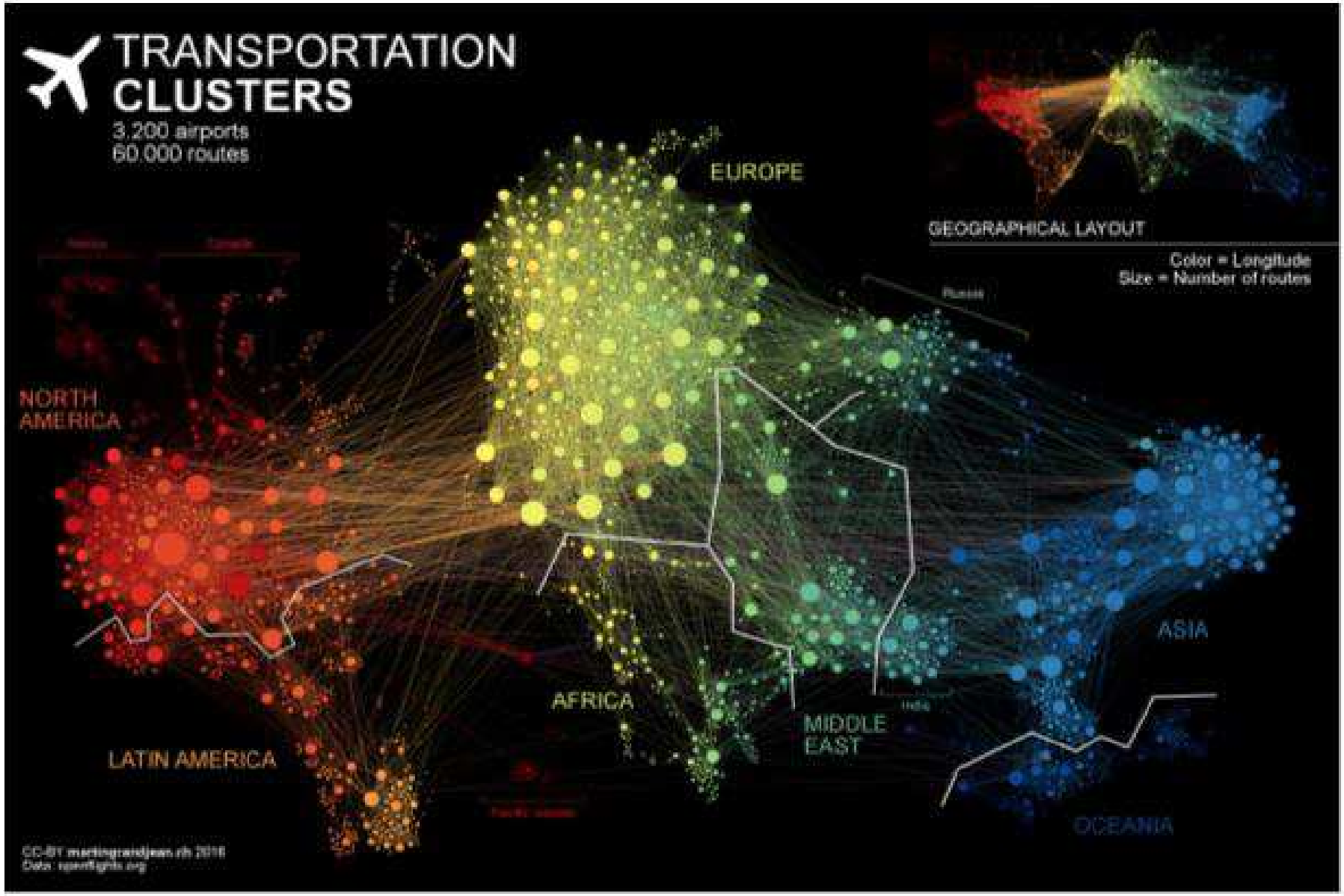
Example 2: protein-protein interaction network



Example 3: co-authors network

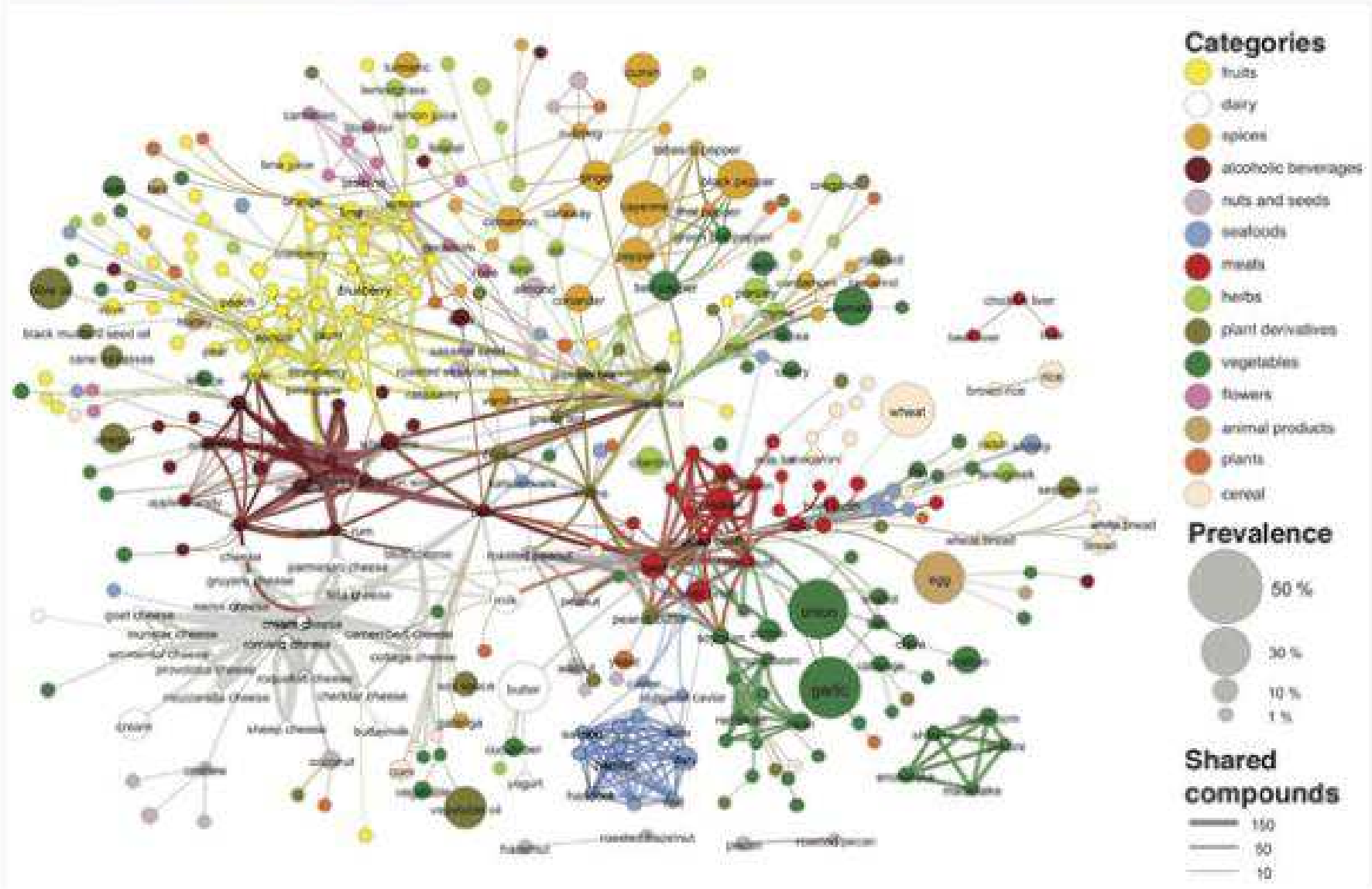


Example 4: International Airport Traffic



Example 5: The Flavor Network

Figure 2: The backbone of the flavor network.



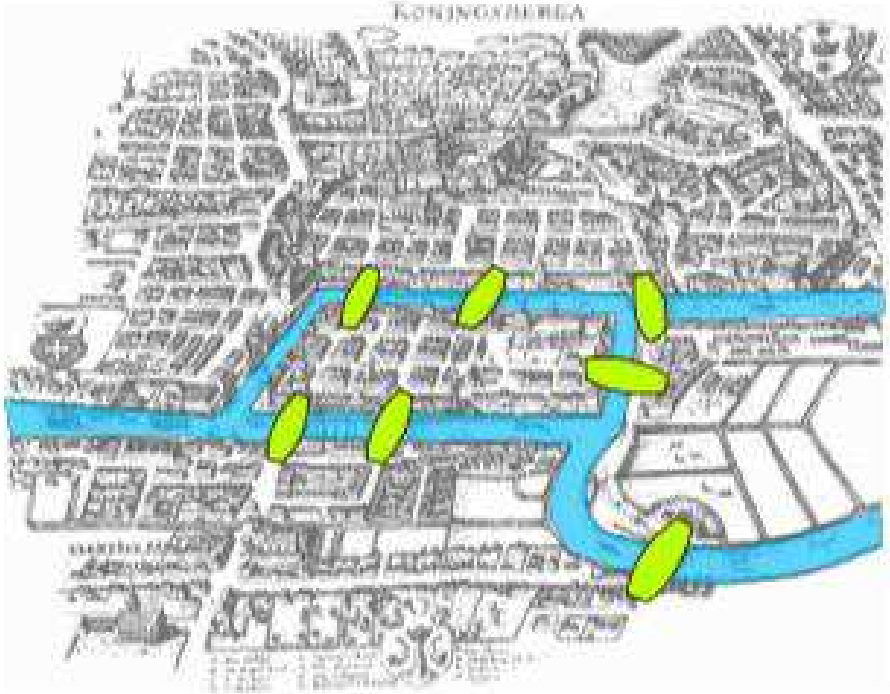
Which ingredients would go well together?

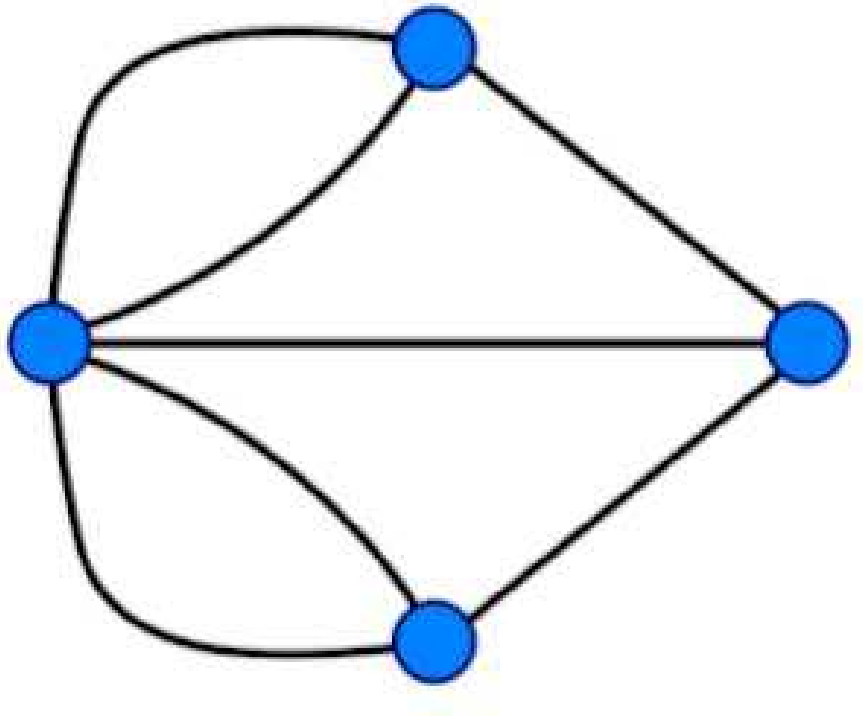
Graphs Basics

The Graph theory starts

Context : The town of Königsberg is built around two islands connected by a bridge, and there exists 6 other bridges to connect these two islands.

Question How to walk through the city that would cross each bridge once and only once ?



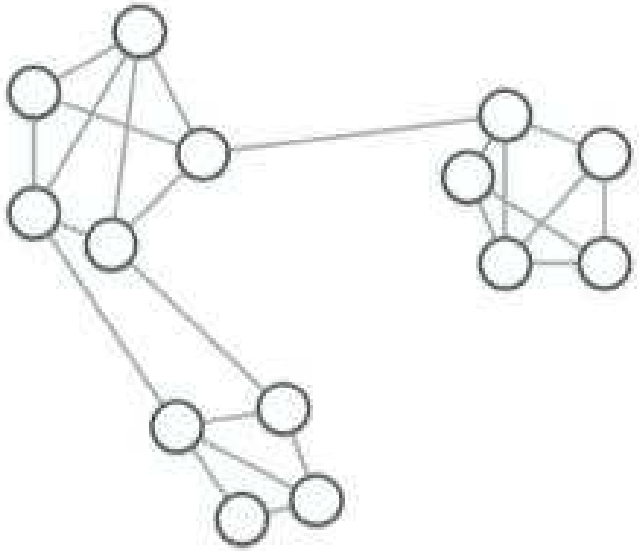


The Graph Theory Paradigms

A graph is denoted by $G = (V, E)$, where

- V is the finite set of vertices (nodes)
- E is the finite set of edges (arcs)

example: {a, b} is the edge connecting vertices a and b

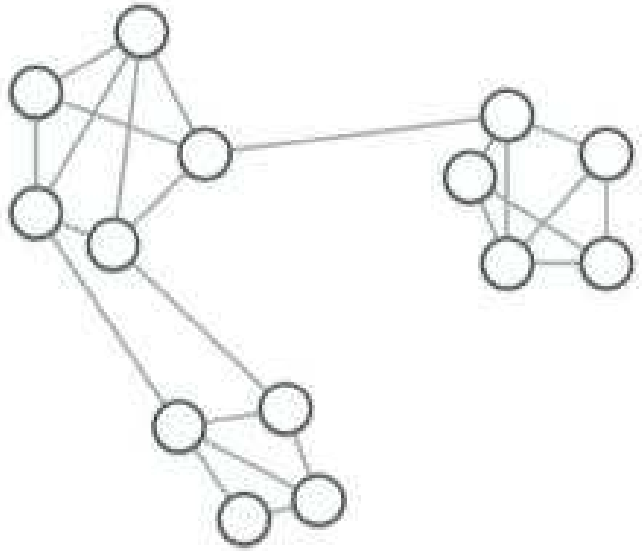


- What is the number of nodes?
- How many edges do we have here?

A graph is denoted by $G = (V, E)$, where

- V is the finite set of vertices (nodes)
- E is the finite set of edges (arcs)

example: {a, b} is the edge connecting vertices a and b



- What is the number of nodes?
- How many edges do we have here?

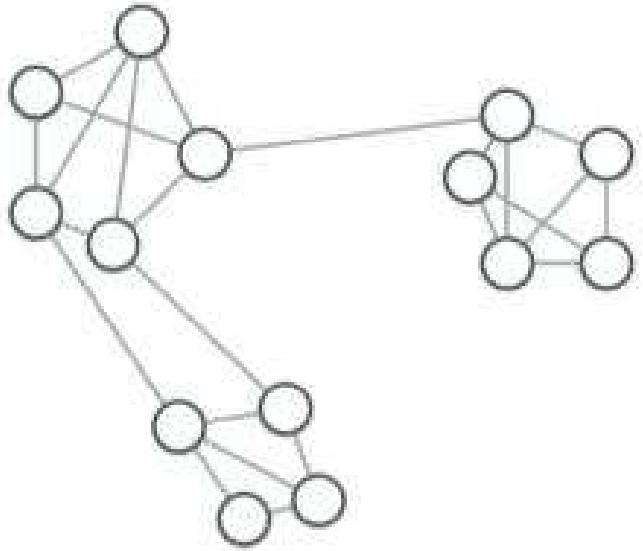
$|V| = 14$ and $|E| = 23$

The Graph Theory Paradigms

A graph is denoted by $G = (V, E)$, where

- V is the finite set of vertices (nodes)
- E is the finite set of edges (arcs)

example: {a, b} is the edge connecting vertices a and b



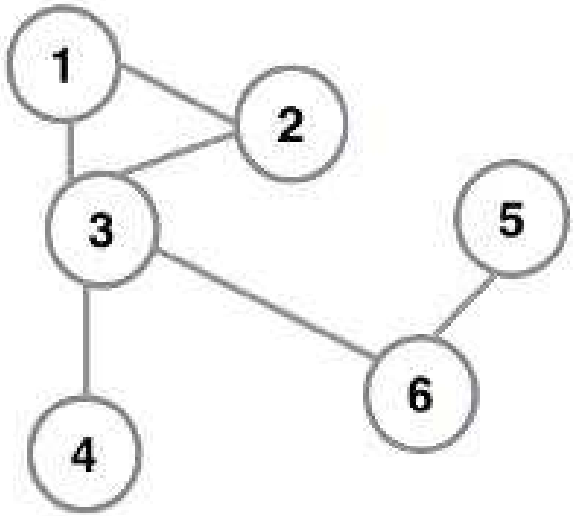
- What is the number of nodes?
- How many edges do we have here?

  $|V| = 14$ and $|E| = 23$
- $|V| = n$ is called the order of the graph G
- $|E| = m$ is called the size of the graph G
- If $|E| = n(n-1)/2$, the graph is complete.

The Adjacency Matrix

A graph can be represented by an adjacency matrix, that indicates the connexion between the nodes.

The adjacency matrix is a 2D array of size $n \times n$ where n is the number of vertices in a graph.



where

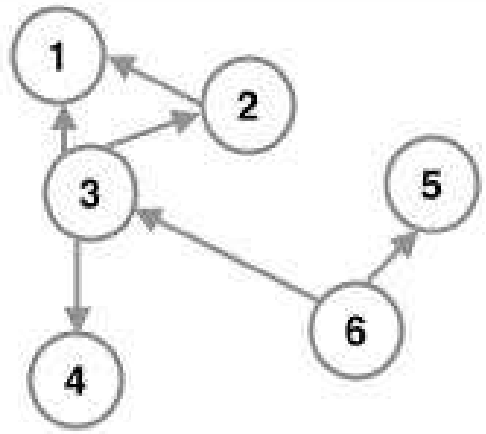
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } v_j \text{are connected} \\ 0 & \text{otherwise.} \end{cases}$$

NB: in this case the adjacency matrix is symmetric

Other Types of Graphs

Directed graphs

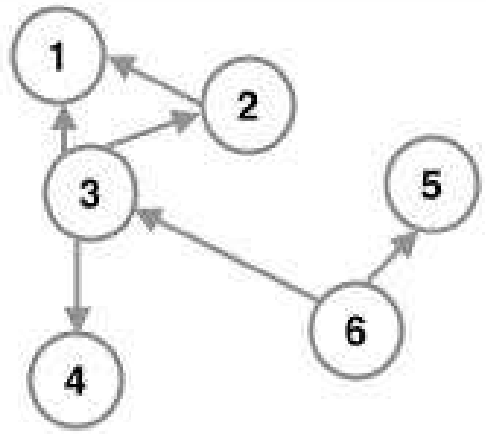


NB: A is no more symmetric

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

Other Types of Graphs

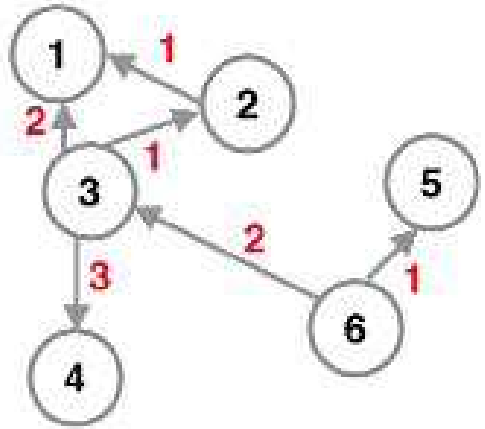
Directed graphs



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

NB: A is no more symmetric

Weighted graphs

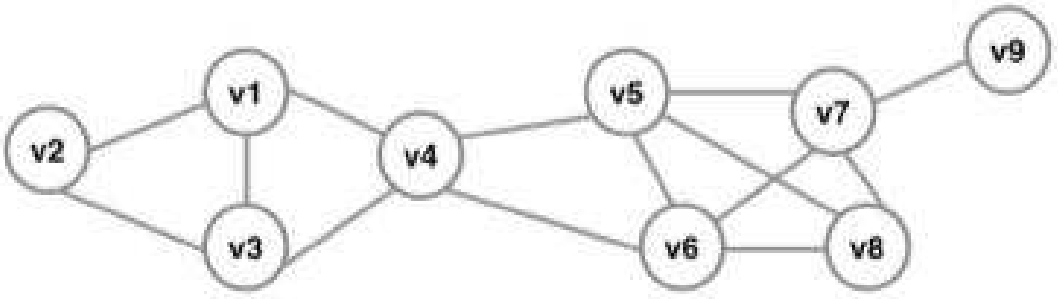


$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 \end{bmatrix}$$

$$a^{ij} = \omega(v^i, v^j) \rightarrow A = W = (w^{ij}) = (\omega(v\,, v\,))$$

Key Notions : Definitions and Properties (1)

Consider the following labeled graph $G = (V, E)$, where $|V| = 9$ and $|E| = 14$



Path : sucession of vertices and edges that allow to go from one vertex to another. A path $P = (V(P), E(P))$ is a subgraph such that

$$V(P) = \{v^{i0}, \ldots, v^{ik}\} \text{ and } E(P) = \{\{v^{i_0}, v^{i_1}\}, \{v^{i_1}, v^{i_2}\}, \ldots, \{v^{i_{k-1}}, v^{i}_{k}\}\}.$$

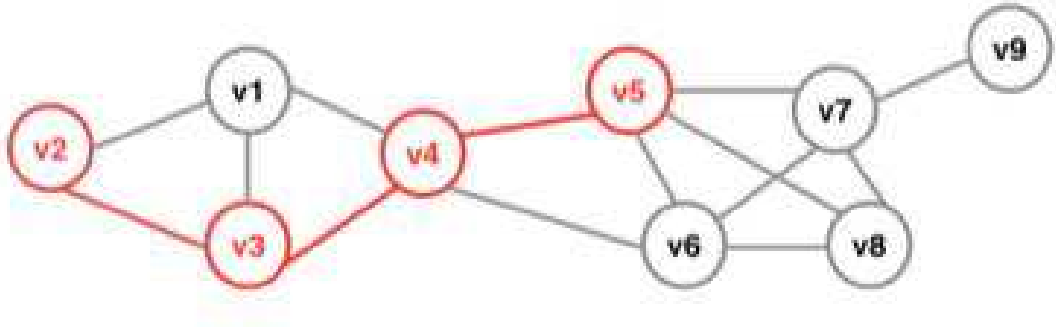
Geodesic distance δ_{ij} : number of edges of the shortest path between two vertices.

Ex: from v_2 to v_5 several paths are possible, and there exists a shortest path where $\delta_{ij} = 3$.

Connected graph : iff $\forall u, v$, there is at least one path connecting v and u. Is the graph presented above connected?

Key Notions : Definitions and Properties (1)

Consider the following labeled graph $G = (V, E)$, where $|V| = 9$ and $|E| = 14$



Path : sucession of vertices and edges that allow to go from one vertex to another. A path $P = (V(P), E(P))$ is a subgraph such that

$$V(P) = \{v^{i0}, \ldots, v^{ik}\} \text{ and } E(P) = \{\{v^{i_0}, v^{i_1}\}, \{v^{i_1}, v^{i_2}\}, \ldots, \{v^{i_{k-1}}, v^{i}_{k}\}\}.$$

Geodesic distance δ_{ij} : number of edges of the shortest path between two vertices.

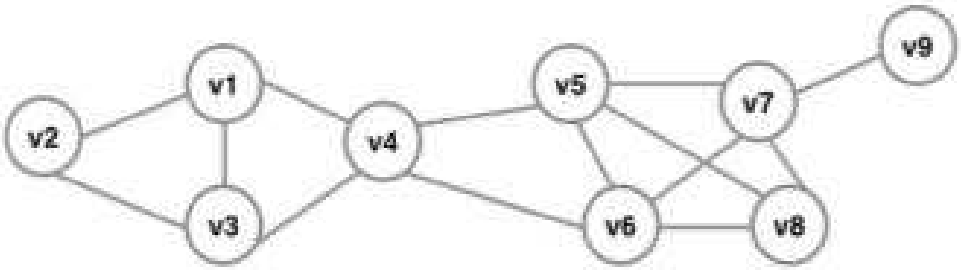
Ex: from v_2 to v_5 several paths are possible, and there exists a shortest path where $\delta_{ij} = 3$.

Connected graph : iff $\forall u, v$, there is at least one path connecting v and u. Is the graph presented above connected?

Key Notions : Definitions and Properties (2)

Density of a graph : ratio between the number of edges observed (size) $|E|$ and the maximum number of possible edges.

$$Density = \frac{|E|}{|V|\frac{(|V|-1)}{2}}$$



Key Notions : Definitions and Properties (2)

Density of a graph : ratio between the number of edges observed (size) $|E|$ and the maximum number of possible edges.

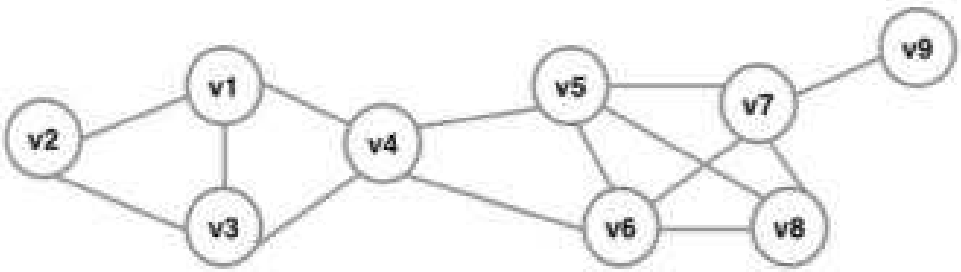
$$Density = \frac{|E|}{|V|\frac{(|V|-1)}{2}}$$

Diameter of a graph : The greatest geodesic distance possible between 2 vertices in the graph

Neighborhood N_i of a node: v_i is a subgraph of G induced by all vertices adjacent to v_i. $Ex : N_3 = \{2, 1, 4\}$.

Connection strengths between two vertices: can be evaluated from the importance of overlap between their neighborhoods. The higher the recovery, the stronger the link.

$$overlap(v_i, v_j) = \frac{|N^i \cap N^j|}{|N^i \cup N^j| - 2}$$



TO DO

1. Diameter of this graph ?
2. Strength between v_1 and v_4 ? v_5 and v_4 ? Conclusion?

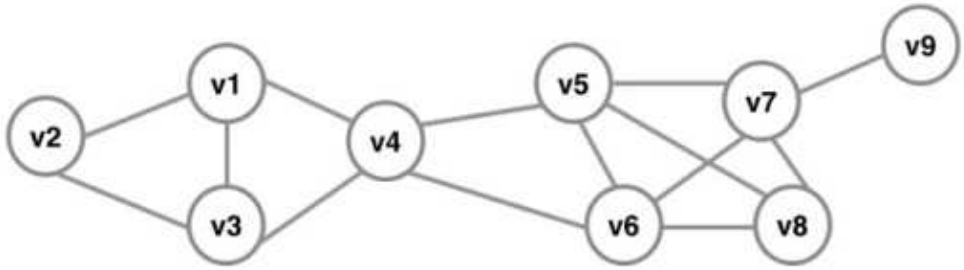
Key Notions : Definitions and Properties (3)

Degree centrality of a node (in terms of neighborhood) : number of edges associated with it (number of adjacent neighbors) absolute $C_D(v_i)$ or relative $C'_D(v_i)$.

$$C_D(v_i) = d_i = \sum_j A_{ij}$$

$$C'_D(v_i) = \frac{C_D(v_i)}{n-1}$$

$Ex : C_D(v3) = 3, C'_D(v_3) : 3/8$



Average centrality : average distance from a given node to all the other ones.

$$C_{avg}(v_i) = \frac{1}{n-1}\sum_{j \neq i} \delta_{ij}$$

Closeness centrality : inverse of the average degree.

$$C_C(v_i) = \frac{1}{C_{avg}(v_i)}$$

Community Detection

What is a community ?

Community (social) : group of people who have special ties because they have particular affinities, or have similar characteristics, or share interests.

What is a community ?

Community (social) : group of people who have special ties because they have particular affinities, or have similar characteristics, or share interests.

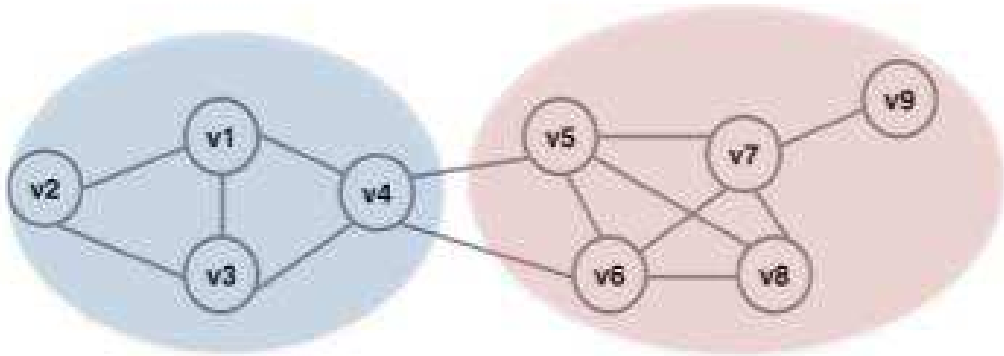
Community (graph) : a set nodes that are strongly linked to each other, and weakly linked with nodes outside the community.

→ implies that it must exists at least a path between two nodes of a community, and this path must be internal to the community.

Clustering: in the clustering framework, a community is a cluster of nodes in a graph → there are no ground-truth labels available !

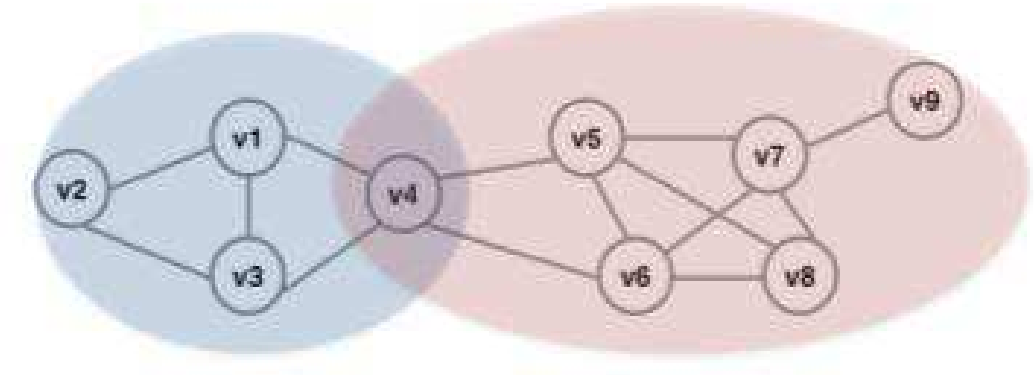
Challenges :

1. Define a community?
2. How many communities?
3. Evaluate the quality?



Disjoint communities vs. overlapping communities

Just like in clustering, the partition can be crisp (disjoint communities) or fuzzy (overlapping communities).
Membership in a community is not necessarily unambiguous!

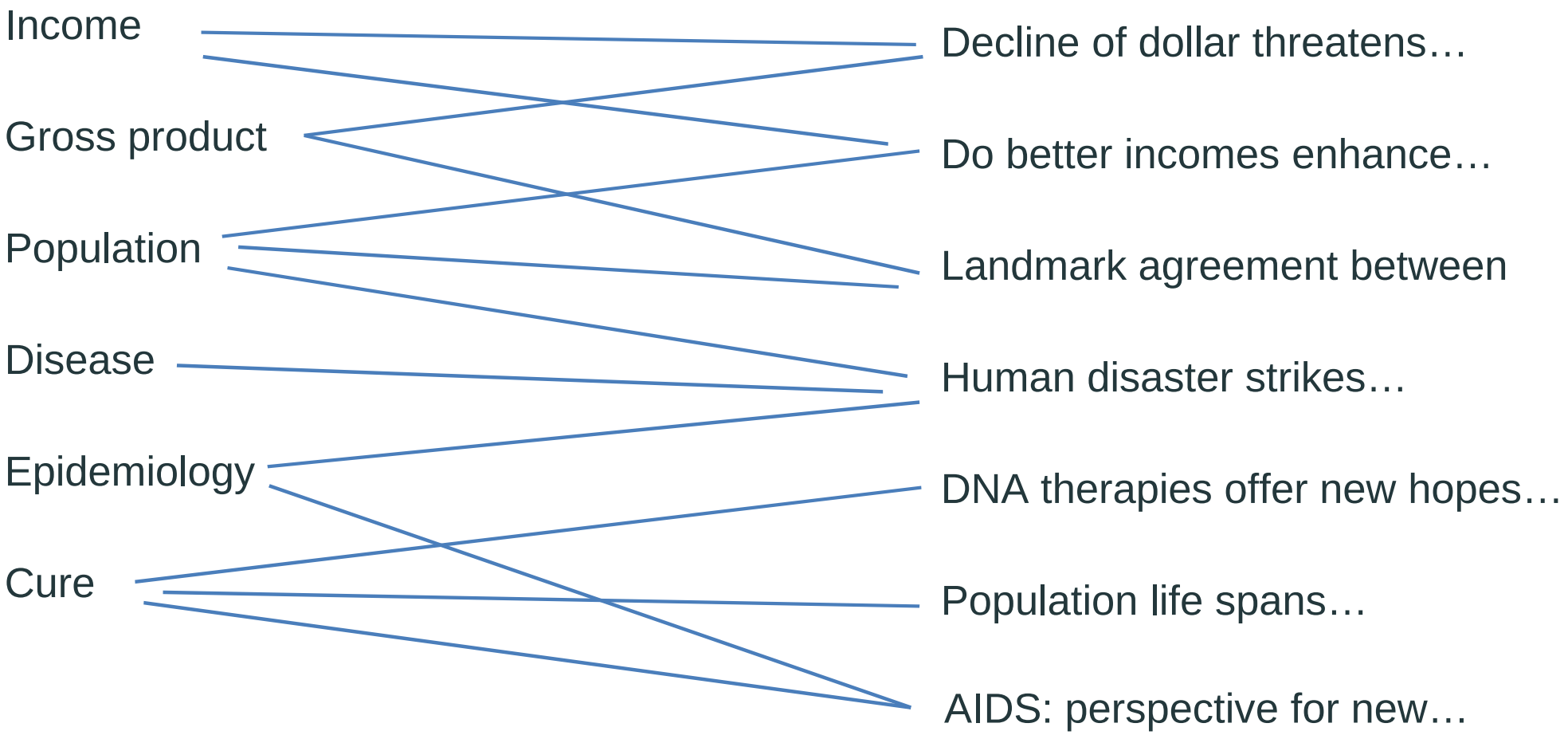


Ex: node v_4 belongs to both communities, acting as an interface between both. In this case, we perceive the notion of centrality and influence.

For the rest of the course, we focus only on disjoint communities.

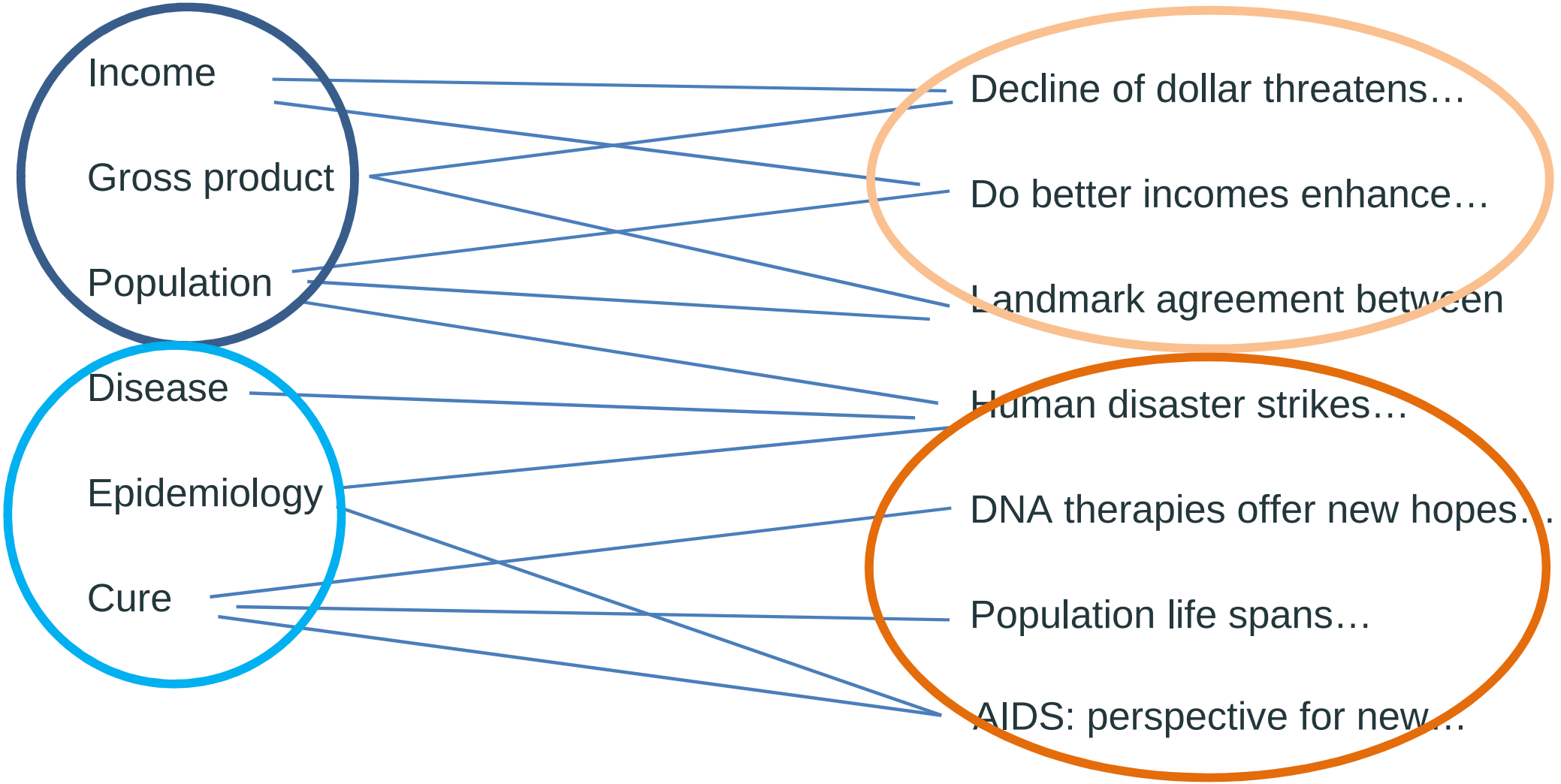
Possible applications to document analysis

Possible applications to document analysis



Clustering of documents sharing the same words / topics (of topic / words present in the same documents)

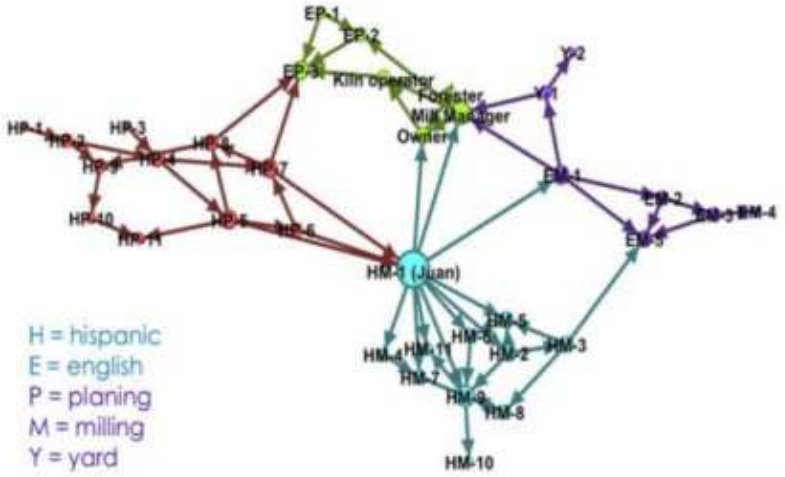
Possible applications to document analysis



Clustering of documents sharing the same words / topics (of topic / words present in the same documents)

Interest of community detection - an example

The example of the sawmill network (Michael and Massey, 1997)



(Selmane, 2015)

The employees were from different ethnic groups (*E, H*), and assigned to different tasks (*P, M, Y*).

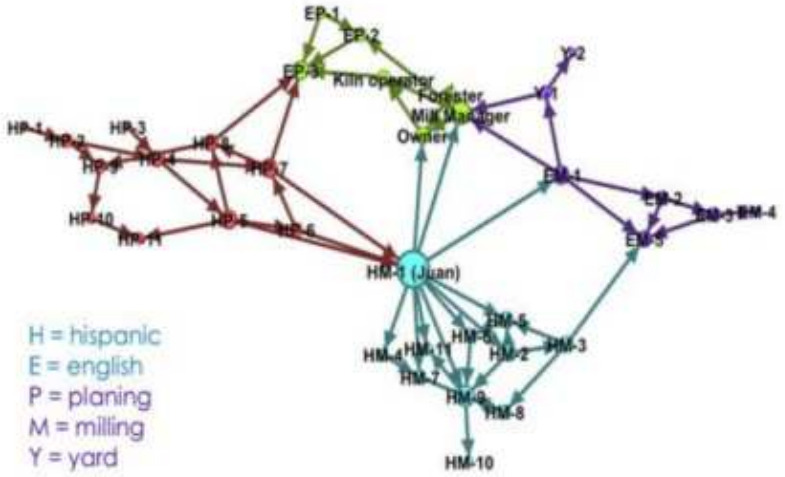
When a change of organization was proposed, strong opposition emerged.

↓

A communications map, based on the frequency of discussions between employees was produced.

Interest of community detection - an example

The example of the sawmill network (Michael and Massey, 1997)



(Selmane, 2015)

The employees were from different ethnic groups (*E, H*), and assigned to different tasks (*P, M, Y*).

When a change of organization was proposed, strong opposition emerged.

↓

A communications map, based on the frequency of discussions between employees was produced.

Juan was playing a central role in the network. To make the changes, they persuaded him first, and then the acceptance just spread to others later.

Modularity [Newman and Girvan]

Modularity is a frequently used quality function.

Idea: a random graph is not expected to have a community structure

→ compare the actual density of edges in a subgraph and a random subgraph

Modularity: difference between the number of connections observed and the theoretical number (expected) obtained if the connections were distributed randomly(1) between the nodes. For a given community k

$$Q(k) = \sum_{i,j \in k} A_{ij} - \frac{d_i d_j}{2m},$$

where m is the number of observed connections, d_i and d_j are the centrality degrees of vertices v_i and v_j, respectively.

Modularity of the network after the partitioninng into K clusters

$$Q = \frac{1}{2m} \sum_{k=1}^{K} \sum_{i,j \in k} \left(A_{ij} - \frac{d_i d_j}{2m} \right)$$

→ The greater the modularity, the better the partition

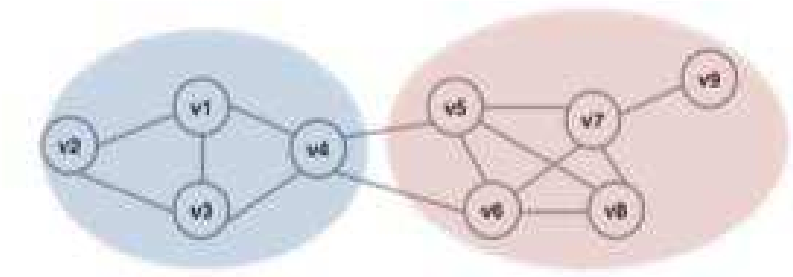
Modularity - the matrix formulation

One can write the modularity as follows

$$Q = \frac{1}{2m} Tr(S^{\mathsf{T}} BS),$$

where

- S is a $(n \times K)$ indicator matrix associating each node to one of the community
- B is the $(n \times n)$ modularity matrix, i.e. $B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$.



$$S = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$Q \approx 0.35$$

The modularity can be used to compare partitions obtained with different algorithms and/or to detect the number of communities.

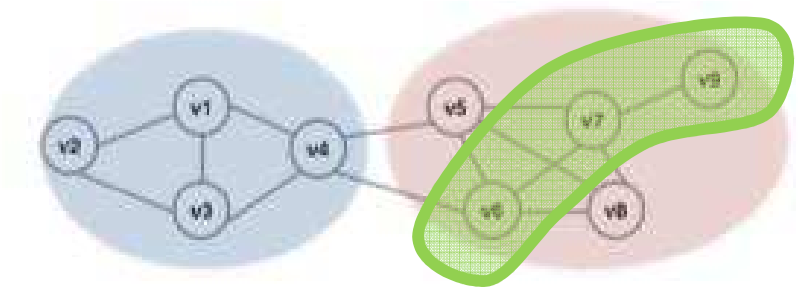
Modularity - the matrix formulation

One can write the modularity as follows

$$Q = \frac{1}{2m} Tr(S^{\mathsf{T}} BS),$$

where

- S is a $(n \times K)$ indicator matrix associating each node to one of the community
- B is the $(n \times n)$ modularity matrix, i.e. $B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$.



$$S = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

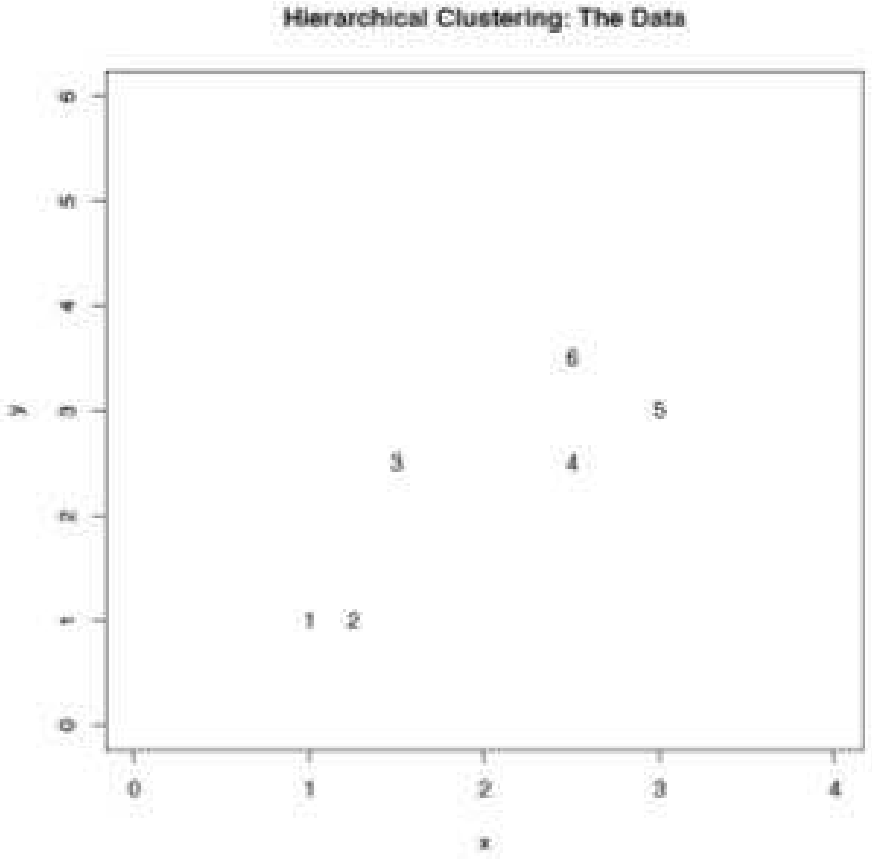
$$Q \approx 0.06$$

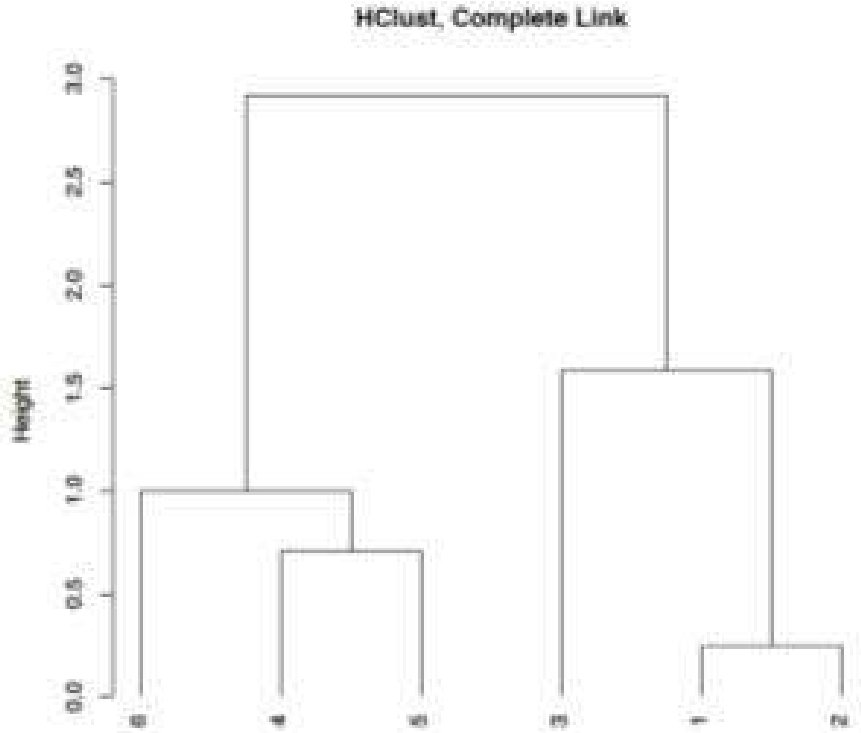
The modularity can be used to compare partitions obtained with different algorithms and/or to detect the number of communities.

Outline

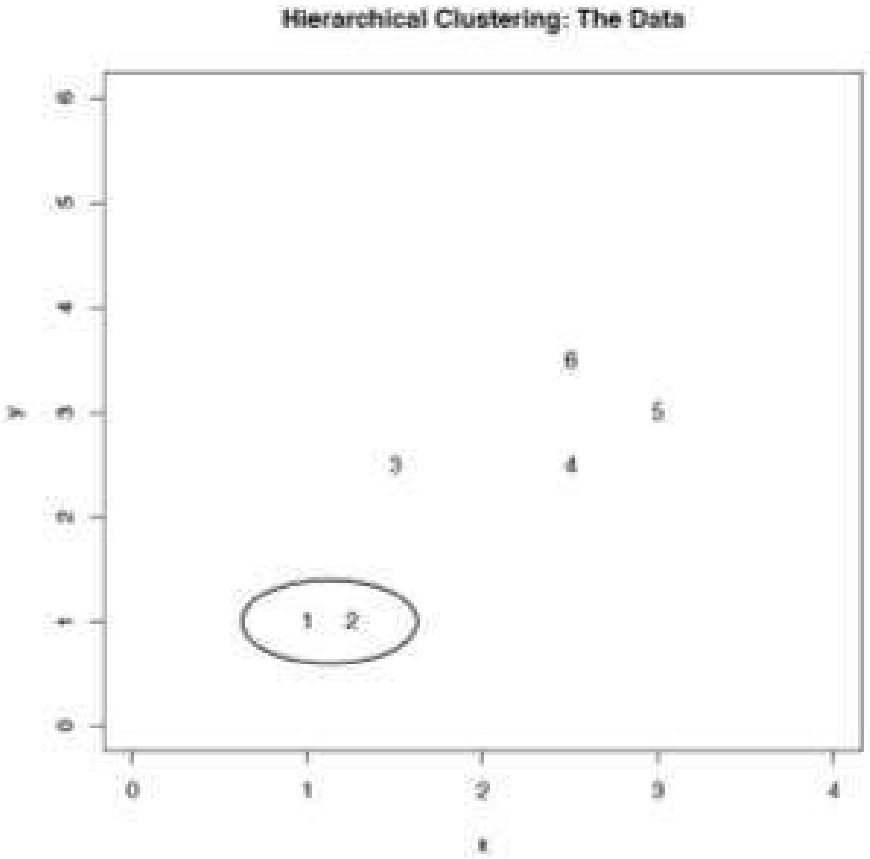
Introduction

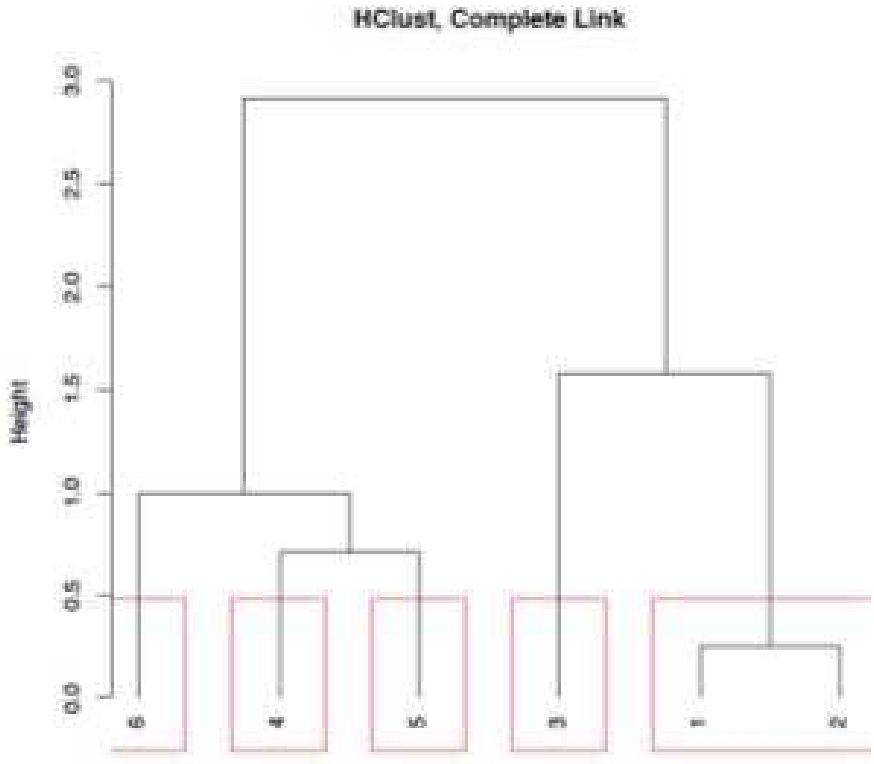
At the starting point, the *n* objects to cluster are their own classes, then at each stage we merge the two most similar clusters.





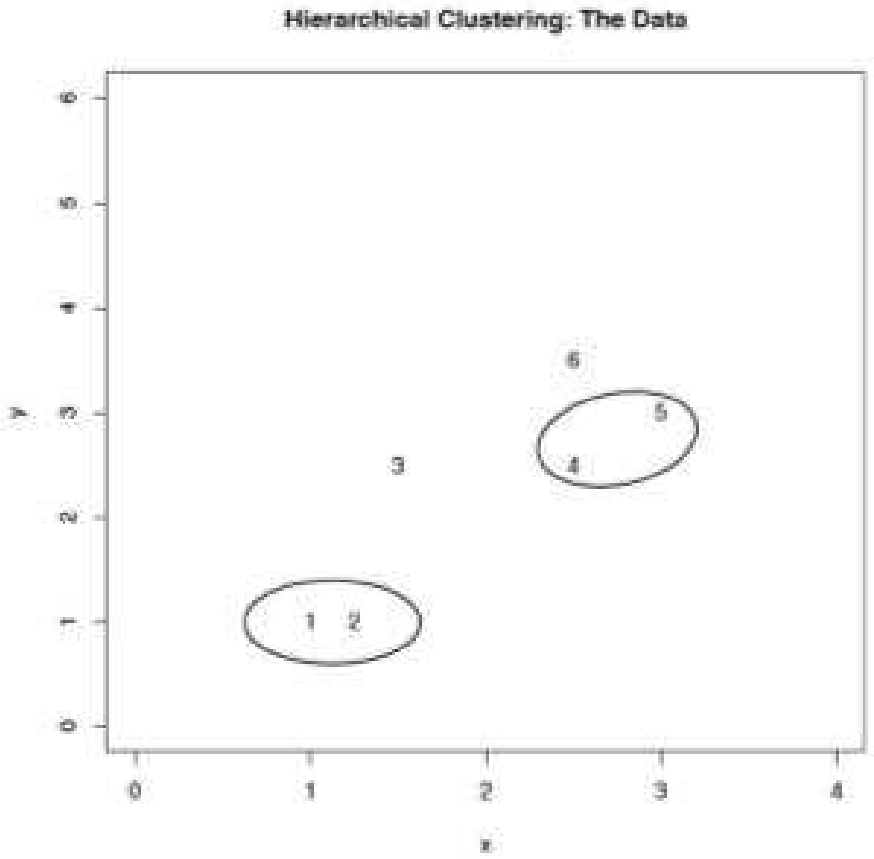
At the starting point, the *n* objects to cluster are their own classes, then at each stage we merge the two most similar clusters.

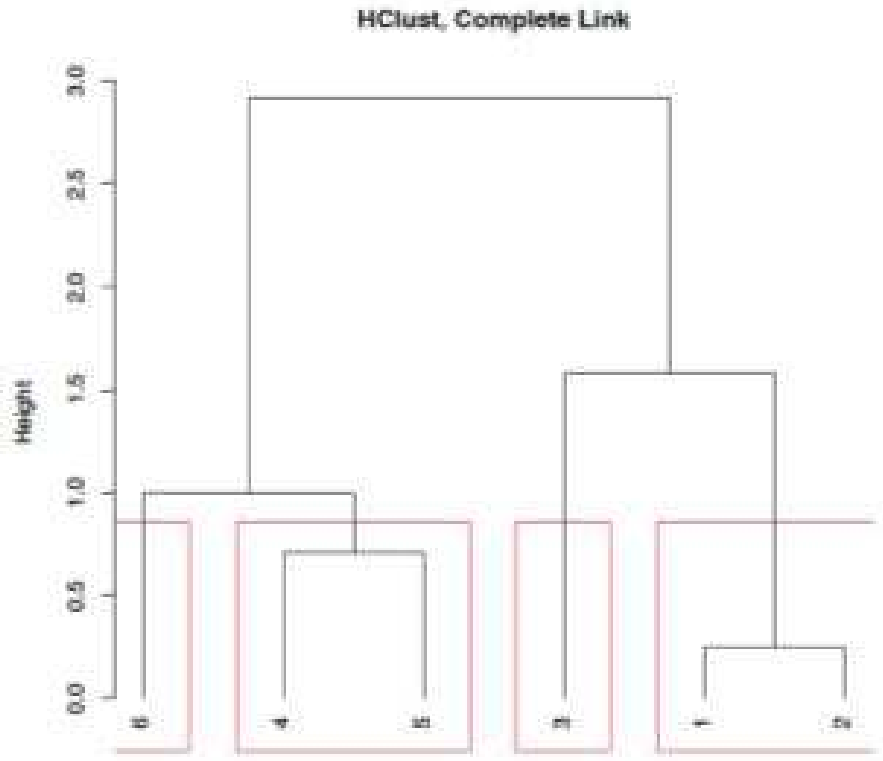




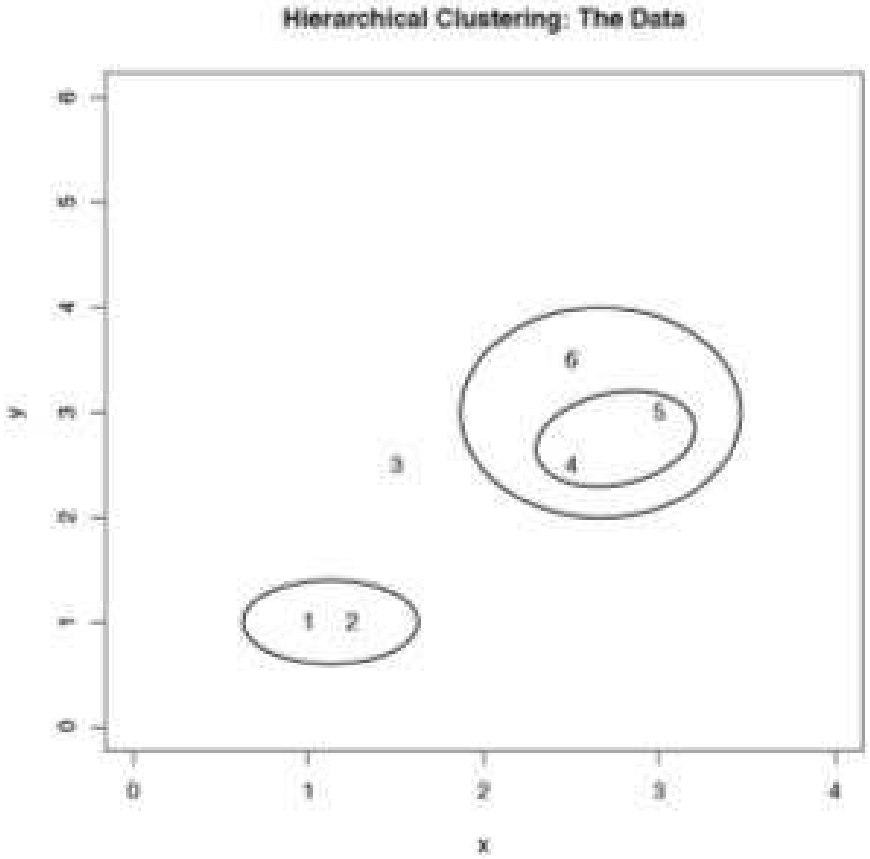
Introduction

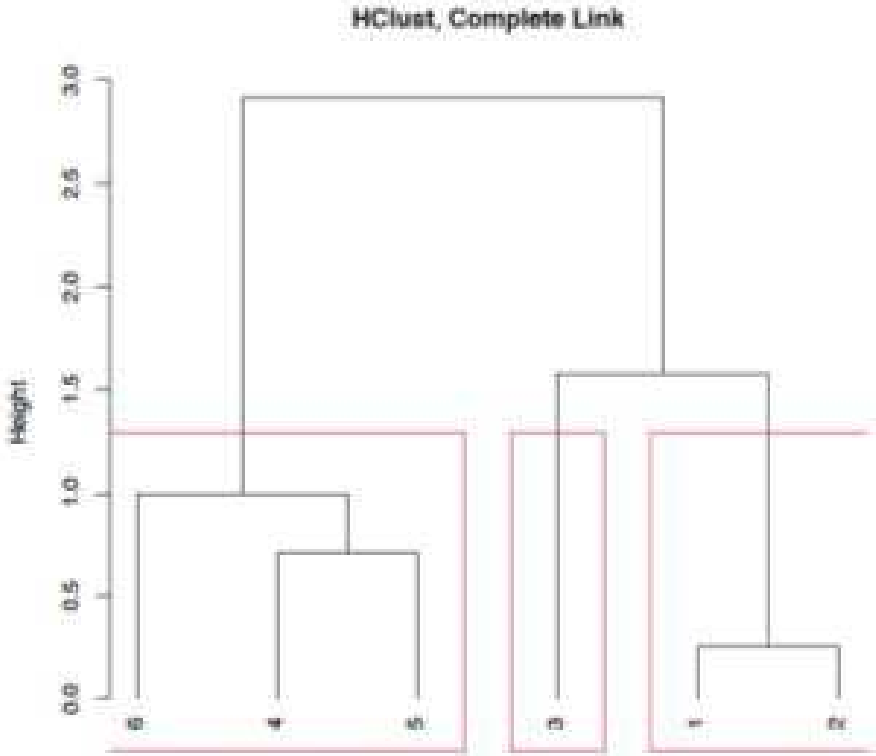
At the starting point, the *n* objects to cluster are their own classes, then at each stage we merge the two most similar clusters.



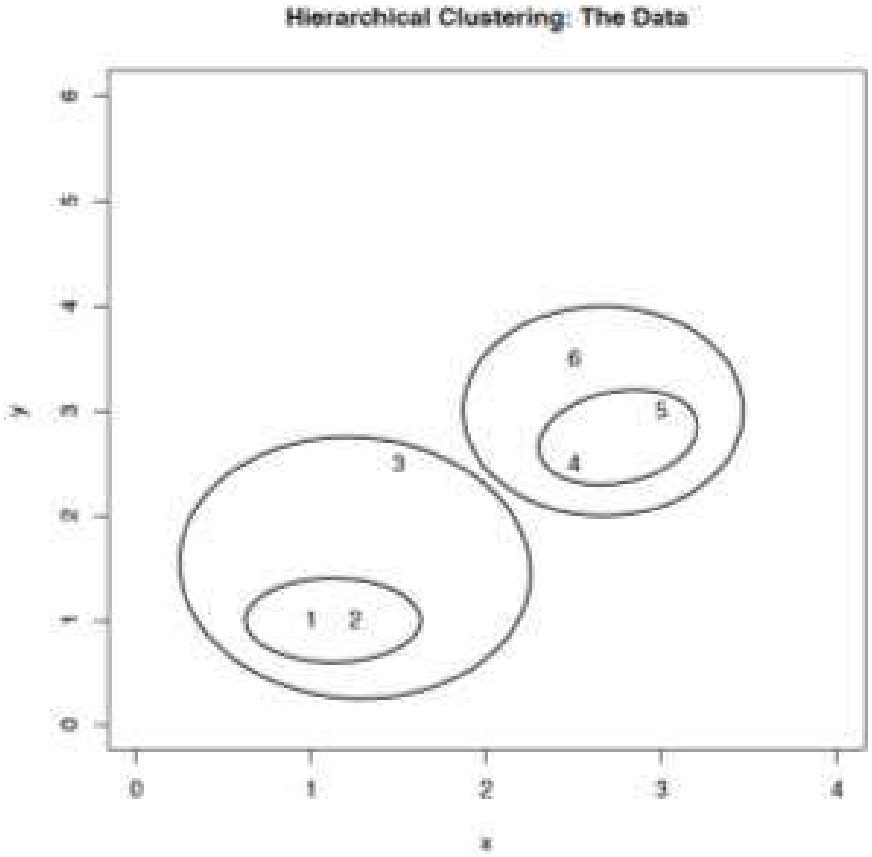


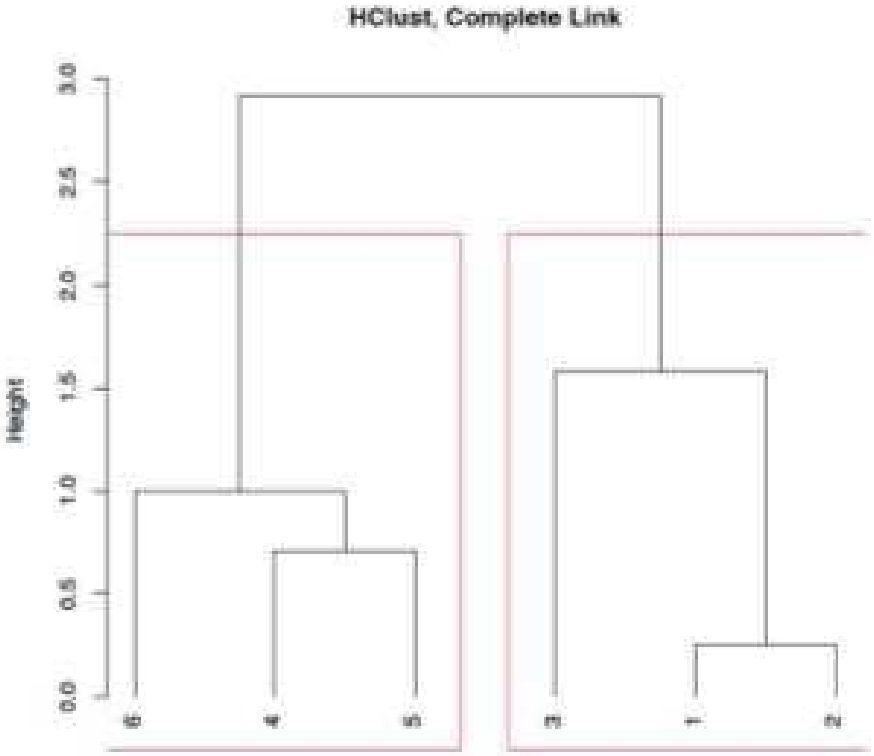
At the starting point, the *n* objects to cluster are their own classes, then at each stage we merge the two most similar clusters.



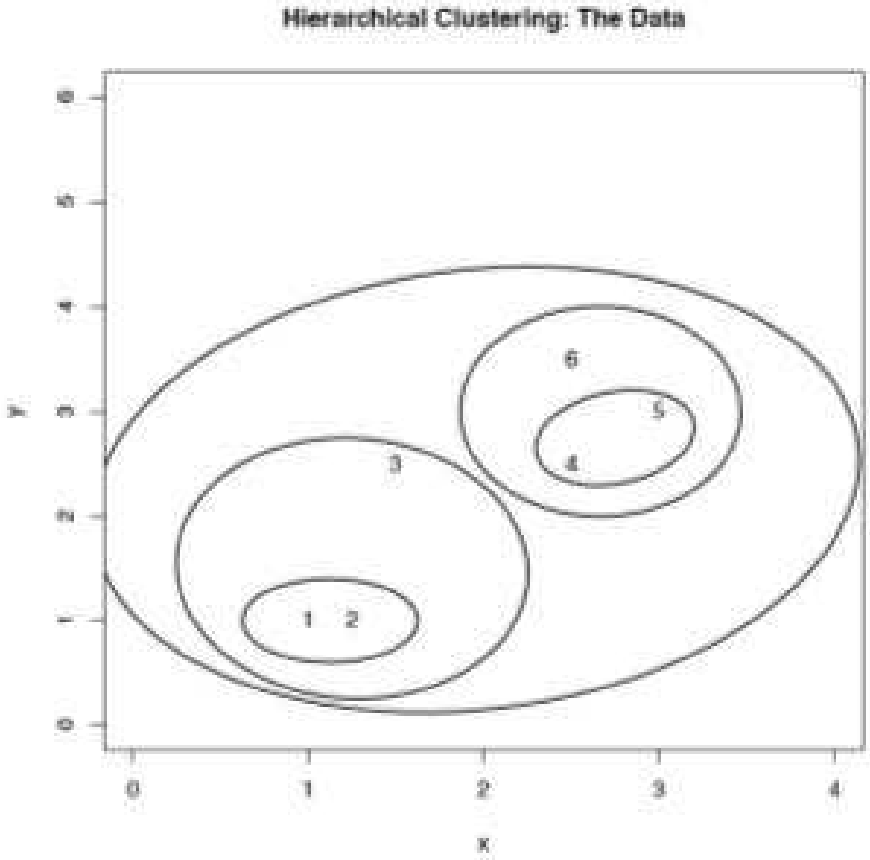


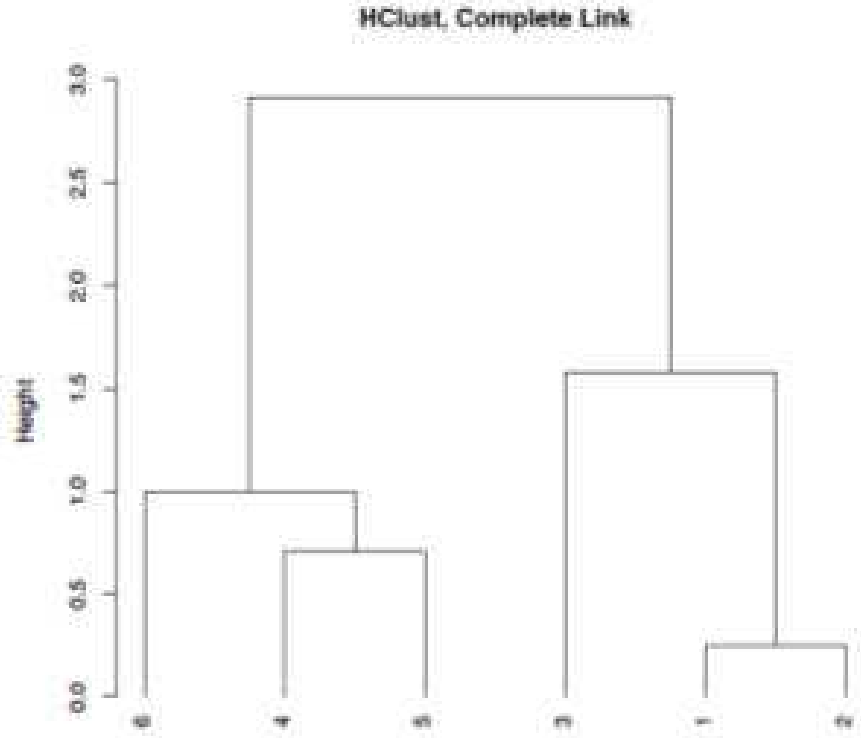
At the starting point, the *n* objects to cluster are their own classes, then at each stage we merge the two most similar clusters.





At the starting point, the *n* objects to cluster are their own classes, then at each stage we merge the two most similar clusters.

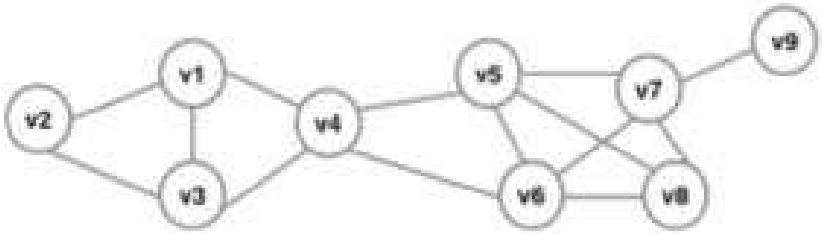




Agglomerative Hierarchical Algorithms - the distance / similarity measures

Similarity: in graph theory, the similarity is defined in terms of neighbourhood. Two nodes are close if there is a high overlapping between their neighborhoods.

Example of indices / distance



- The Jaccard index

$$Jaccard(v_i, v_j) = \frac{|N^i \cap N^j|}{|N^i \cup N^j|}$$

$$0 \leq Jaccard(v^i, v^j) \leq 1$$

- Cosine similarity

$$Cosine(v_i, v_j) = \frac{|N^i \cap N^j|}{\sqrt{|N^i| \, |N^j|}}$$

$$0 \leq Cosine(v^i, v^j) \leq 1$$

- $Jaccard(v^4, v^5) = ?$
- $Cosine(v^4, v^5) = ?$
- $overlap(v^4, v^5) = 1/5$

N.B. Dissimilarity: any decreasing function of sim.

Agglomerative Hierarchical Algorithms - the link choice

For a given dissimilarity measure *dis* between two nodes, several dissimilarities between clusters D exist:

- the single linkage :

$$D(A, B) = min\{dis(x, y) : x \in A, y \in B\}$$

- the complete linkage :

$$D(A, B) = max\{dis(x, y) : x \in A, y \in B\}$$

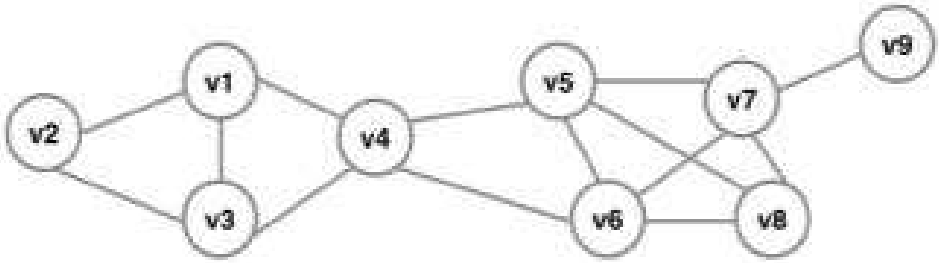
- the average linkage :

$$D(A, B) = \frac{1}{|A|.|B|} \sum_{x \in A} \sum_{y \in B} dis(x, y)$$

Two main phases:

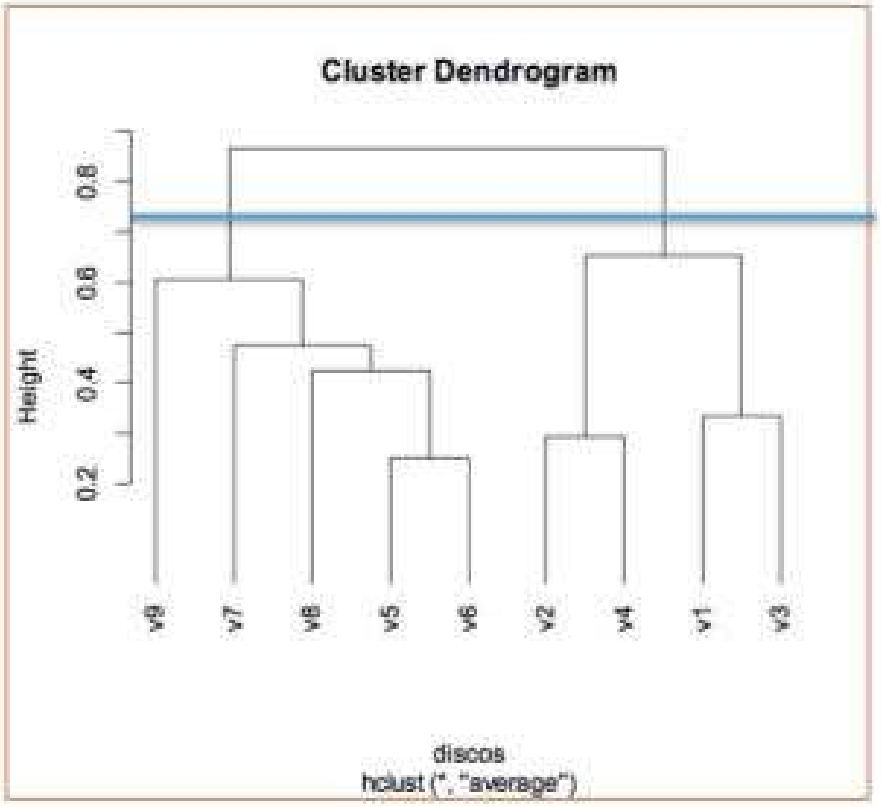
1. Initialiazation of the algorithm: compute the pairwise distances between each pair of nodes
2. Iterations over the following steps :
   - Group 2 elements (nodes or groups) based on the chosen *dis*
   - Update the pairwise distance matrix between groups

Example



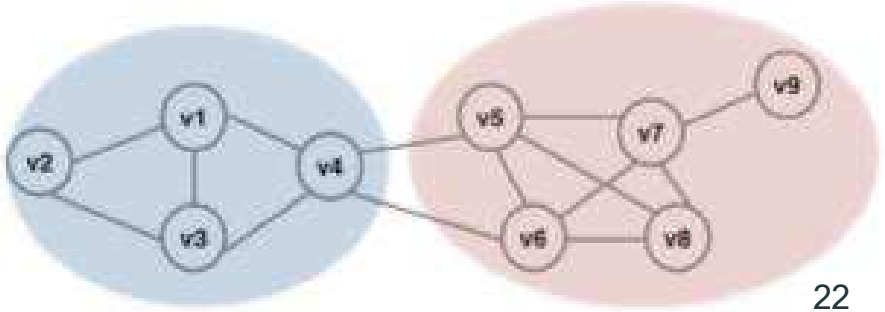
	V1	V2	V3	V4	V5	V6	V7	V8	V9
v1	1	0.408	0.667	0.289	0.289	0.289	0.000	0.000	0.000
v2	0.408	1	0.408	0.707	0.000	0.000	0.000	0.000	0.000
v3	0.667	0.408	1	0.289	0.289	0.289	0.000	0.000	0.000
v4	0.289	0.707	0.289	1	0.250	0.250	0.500	0.577	0.000
v5	0.289	0.000	0.289	0.250	1	0.750	0.500	0.577	0.500
v6	0.289	0.000	0.289	0.250	0.750	1	0.500	0.577	0.500
v7	0.000	0.000	0.000	0.500	0.500	0.500	1	0.577	0.000
v8	0.000	0.000	0.000	0.577	0.577	0.577	0.577	1	0.577
v9	0.000	0.000	0.000	0.000	0.500	0.500	0.000	0.577	1

Similarity matrix: cosine



Dissimilarity matrix: 1-cosine

TO DO: write the first two iterations of the agglomerative algorithms for the given graph with the average linkage



Outline

Edge betweenness

Idea : find the connecting edges to find the communities

Edge betweenness: indicates the frequency with which it is borrowed when one considers the shortest path between each pair of nodes.

The higher the value, the higher the connection is important because it establishes a bridge between groups of vertices.

$$eb(e) = \sum_{i} \sum_{j>i} \frac{\sigma_{ij}(e)}{\sigma_{ij}}$$

where

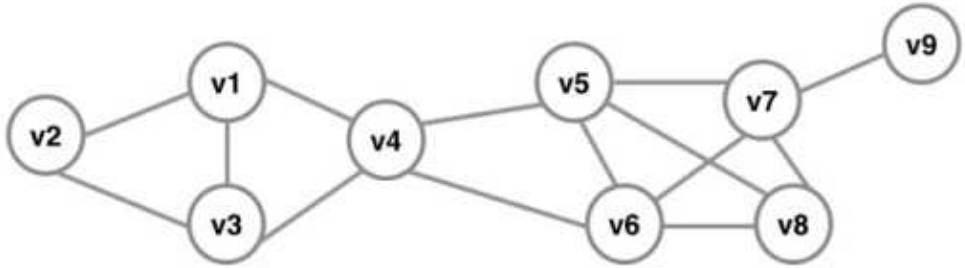
- $\sigma^{ij}(e)$ is number of shortest paths between nodes v^i and v^j going through e.
- σ^{ij} is the number of shortest paths between nodes v^i and v^j

Edge betweenness

Idea : find the connecting edges to find the communities

Edge betweenness: indicates the frequency with which it is borrowed when one considers the shortest path between each pair of nodes.

The higher the value, the higher the connection is important because it establishes a bridge between groups of vertices.



- number of shortest paths between nodes v^i and v^k going through e.
- number of shortest paths between nodes v^i and v^k
- for $(v^i, v^k) = (v^2, v^9)$ and e$= (v^4, v^6)$
- for $(v^i, v^k) = (v^1, v^9)$ and e$= (v^1, v^4)$

Divisive algorithm [Newman and Girvan]

Principle: divide the graph by finding and "removing" edges connecting communities, *i.e.* edges on the maximum of shortest paths between these communities (max edge betweenness).

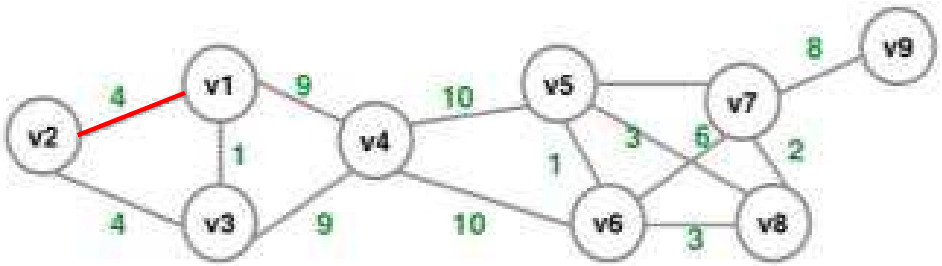
Recursive Algorithm:

1. Compute the edge betweenness for all edges of the running graph
2. Remove the edge with the maximum *eb*
   - if the removing partitions the graph then divide the graph into groups and compute the eb of the subgraphs
   - else update the *eb* for the graph.

   Stopping criteria : graph = singleton or quality criteria (e.g. the modularity)

Complexity edge betwneenness on a graph can be computed in $\mathcal{O}(nm)$ for unweighted graphs and in $\mathcal{O}(nm + n^2 \log n)$ for weighted ones.

Example



Computation of $eb(e[1,2]) = 4$?

$$eb(e) = \sum_{i} \sum_{j>i} \frac{\sigma_{ij}(e)}{\sigma_{ij}}$$

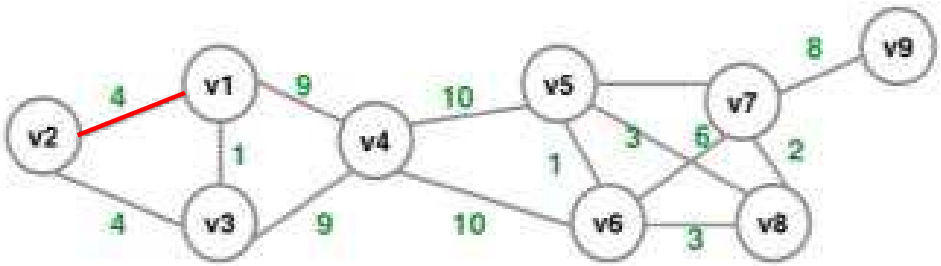
$$1 \longrightarrow (2,3,\ldots) : \frac{1}{1} + \frac{0}{1} + \frac{0}{1} + \ldots = 1$$

$$2 \longrightarrow (3,4,\ldots) : 0 + \frac{1}{2} \times 6 = 3$$

$$3 \longrightarrow (4,5,\ldots) : 0$$

$$\ldots : 0$$

Example



Computation of $eb(e[1,2]) = 4$?

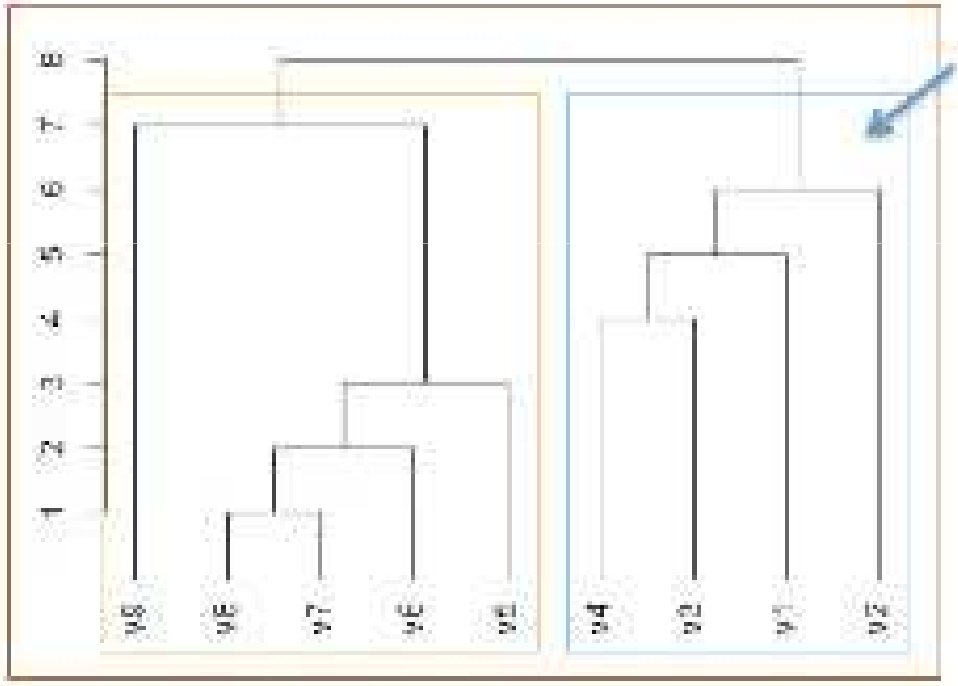
$$eb(e) = \sum_i \sum_{j>i} \frac{\sigma_{ij}(e)}{\sigma_{ij}}$$

$$1 \longrightarrow (2,3,\ldots) : \frac{1}{1} + \frac{0}{1} + \frac{0}{1} + \ldots = 1$$

$$2 \longrightarrow (3,4,\ldots) : 0 + \frac{1}{2} \times 6 = 3$$

$$3 \longrightarrow (4,5,\ldots) : 0$$

$$\ldots : 0$$



Outline

Recall and Intuition

Modularity is a metric that quantifies the quality of an assignment of nodes to communities.

How much more densely connected the nodes within a community are compared to how connected they would be, on average, in a suitably defined random network?

$$Q = \frac{1}{2m} \sum_{k=1}^{K} \sum_{i,j \in k} \left(A_{ij} - \frac{d_i d_j}{2m} \right)$$

Idea: optimizing the modularity, i.e., finding the partitioning of G into K communities that maximizes $\rightarrow$ NP-hard Problem !

Louvain's idea: relies on the gain in modularity ΔQ obtained by moving some node i into a community k.
Observed complexity: generally $O(n \log n)$

The Louvain Algorithm

Initialization: create one community per node.

The Louvain Algorithm

Initialization: create one community per node.

Phase 1: Community assignment

While modularity can be improved: pick a node and compute the change in modularity if it were removed from it's current community and put in a connected community.

- If none of these modularity changes are positive, we keep the node in its current community
- If some of the modularity changes are positive, we move the node into the community for which the modularity change is maximum

Change of Modularity

Phase 1 of the procedure relies on maximising wrt *c*=*out* differences

$$\Delta_{c=in,out} \frac{1}{2m}\left(2k_{i,c} - \frac{2}{2m} k_i \Sigma_{i,c} \right)$$

where

- *c* : community (*i* leaving *out* and entering *in*)
- $k_{i,c}$ number of neighbours of *i* in *c*

Change of Modularity

Phase 1 of the procedure relies on maximising wrt *c*=*out* differences

$$\Delta_{c=in,out} \frac{1}{2m}\left(2k_{i,c} - \frac{2}{2m} k_i \Sigma_{i,c} \right)$$

where

- *c* : community (*i* leaving *out* and entering *in*)
- $k_{i,c}$ number of neighbours of *i* in *c*

Evaluates the change of modularity by removing *i* from its community and then by moving it into a neighbouring community.

The Louvain Algorithm

Initialization: create one community per node.

Phase 1: Community assignment

While modularity can be improved: pick a node and compute the change in modularity if it were removed from it's current community and put in a connected community.

- If none of these modularity changes are positive, we keep the node in its current community
- If some of the modularity changes are positive, we move the node into the community for which the modularity change is maximum

Phase 2 : Coarse Graining

Define a new, coarse-grained network where the nodes are the communities discovered during Phase 1.

Edge weight between the nodes representing two communities $=$ the sum of the edge weights between the lower-level nodes of each community.

The Louvain Algorithm

Initialization: create one community per node.

Phase 1: Community assignment

While modularity can be improved: pick a node and compute the change in modularity if it were removed from it's current community and put in a connected community.

- If none of these modularity changes are positive, we keep the node in its current community
- If some of the modularity changes are positive, we move the node into the community for which the modularity change is maximum

Phase 2 : Coarse Graining

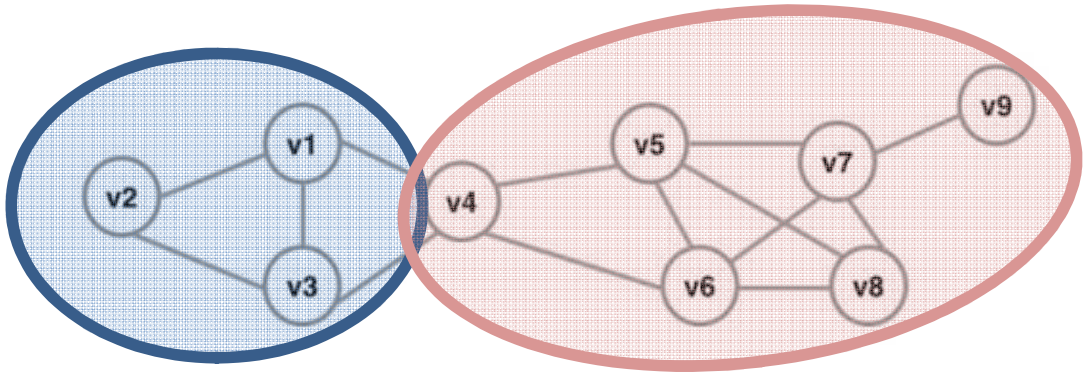
Define a new, coarse-grained network where the nodes are the communities discovered during Phase 1.

Edge weight between the nodes representing two communities $=$ the sum of the edge weights between the lower-level nodes of each community.

Repeat these two phases until only one community remains.

The Louvain algorithm: step

To do: compute the change in modularity induced by switching the community of v_4.



$$\Delta_{c=in,out} \frac{1}{2m}\left(2k_{i,c} - \frac{2}{2m}k_i\Sigma_{i,c}\right)$$

where

- c : community (i leaving *out* and entering *in*)
- $k_{i,c}$ number of neighbours of i in c

Outline

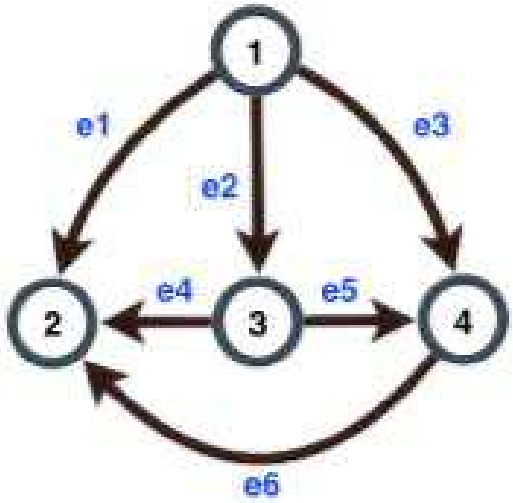
Graph Laplacian (1)

Incidence matrix

each row $\rightarrow$ edge

each column $\rightarrow$ vertex

$$C = C(G) = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$



The Graph Laplacian (1)

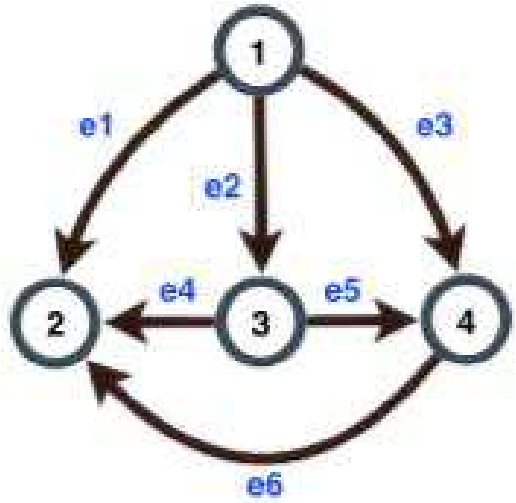
Incidence matrix

each row $\rightarrow$ edge

each column $\rightarrow$ vertex Example: $e1 = e(1,2)$

$\rightarrow C(1,1) = 1$ and $C(1,2) = -1$

$$C = C(G) = \begin{bmatrix} 1 & -1 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$



The Graph Laplacian (1)

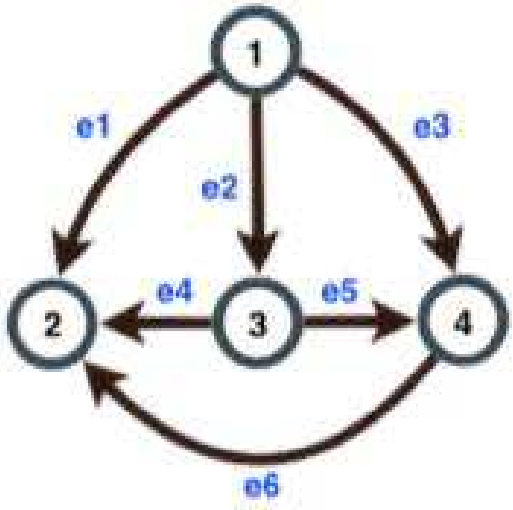
Incidence matrix

each row $\rightarrow$ edge

each column $\rightarrow$ vertex Example: $e1 = e(1,2)$

$\rightarrow$ $C(1,1) = 1$ and $C(1,2) = -1$

$$C = C(G) = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$



Graph Laplacian $L(G) = C^{\mathsf{T}} C$

Graph Laplacian (2)

Symbolicaly, one can write C as a sequence of its n edges

$$C = \begin{bmatrix} e_1^T \\ e_2^T \\ . \\ . \\ . \\ e_n^T \end{bmatrix}$$

Thefore

$$C^T C = \sum_{l=1}^{n} e_l^T e_l$$

Graph Laplacian (2)

Symbolicaly, one can write C as a sequence of its n edges

$$C = \begin{bmatrix} e_1^T \\ e_2^T \\ . \\ . \\ . \\ e_n^T \end{bmatrix}$$

Thefore

$$C^T C = \sum_{l=1}^{n} e_l^T e_l$$

edge(i,j) $\rightarrow \grave{e}$, where $\grave{e} = 1$ at the i-th position and -1 at the j-th position

Consider $\grave{e}\,\grave{e}^T$.

- Gives a matrix with only 4 elements $\neq 0$
- Diagonal count incident edges
- Off-diagonal: existence of an edge
- Lost the direction information

What about the sum?

Graph Laplacian (2)

Symbolicaly, one can write C as a sequence of its n edges

$$C = \begin{bmatrix} e_1^T \\ e_2^T \\ . \\ . \\ . \\ e_n^T \end{bmatrix}$$

Thefore

$$C^T C = \sum_{l=1}^{n} e_l^T e_l$$

What about the sum?

Diagonal counts the degree of each vertex
$\rightarrow$ diagonal matrix D

Off-diagonal marks all the edges
$\rightarrow$ adjacency matrix A

$$C^T C = \sum_{l=1}^{n} e_l^T e_l = D - A$$

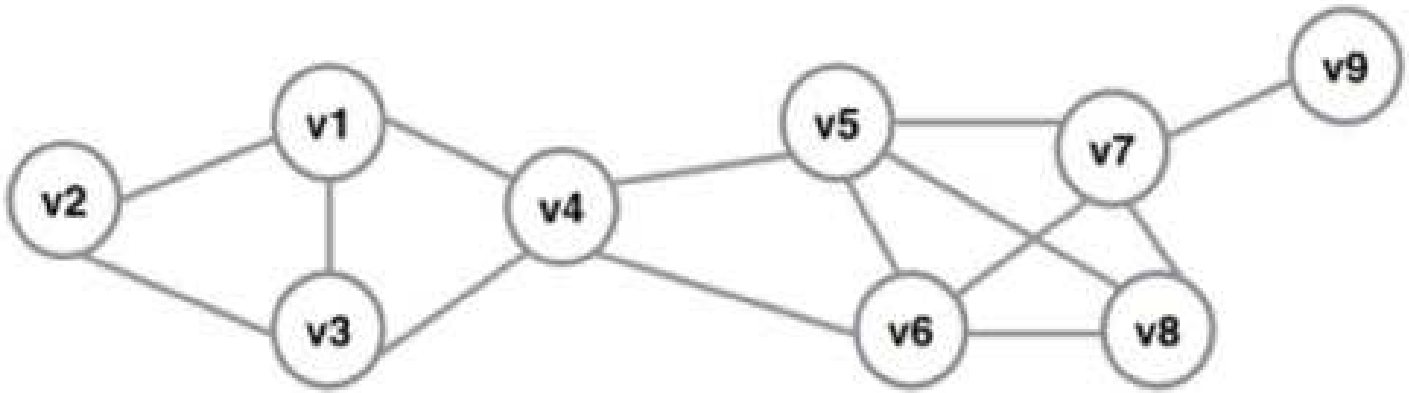
where A is the adjacency matrix of the undirected graph

Frequent use of normalised graph Laplacian

$$D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Example

Compute the Laplacian matrix of the following example



Algebraic Connectivity

Properties of the Graph Laplacian

1. $L(G)$ symmetric
2. $L(G)$ has real-valued, non-negative eigenvalues
   and real-valued, orthogonal eigenvectors
3. G has K connected components iif

$$\lambda^1 = \lambda^2 = \cdots = \lambda^K = 0 \qquad \text{(Fiedler, 1973)}$$

Properties of the Graph Laplacian

1. $L(G)$ symmetric
2. $L(G)$ has real-valued, non-negative eigenvalues
   and real-valued, orthogonal eigenvectors
3. G has K connected components iif

$$\lambda^1 = \lambda^2 = \cdots = \lambda^K = 0 \qquad \text{(Fiedler, 1973)}$$

Recall: let $(\lambda^i, \vec{q}^i)$ be an eigenpair of $L(G)$

$$\Rightarrow L(G)\vec{q}^i = \lambda^i \vec{q}^i$$

Can also be written in a matrix form

$$\Rightarrow L(G)Q = Q\Lambda,$$

where the columns of Q are the eigenvectors and Λ is a diagonal matrix whose diagonal elements are the eigenvalues

Convention: $\lambda^n \geq \lambda^{n-1} \geq \cdots \geq \lambda^2 \geq \lambda^1 \geq 0$

Let consider a graph G with a partition into two communities (+ or -)

- Let V_+ be the set of vertices in +
- Let V_- be the set of vertices in -
- Let $\vec{x}$ be a vector containing the community information

Then the number of cut edges is $\frac{1}{4} x^T L(G) x$

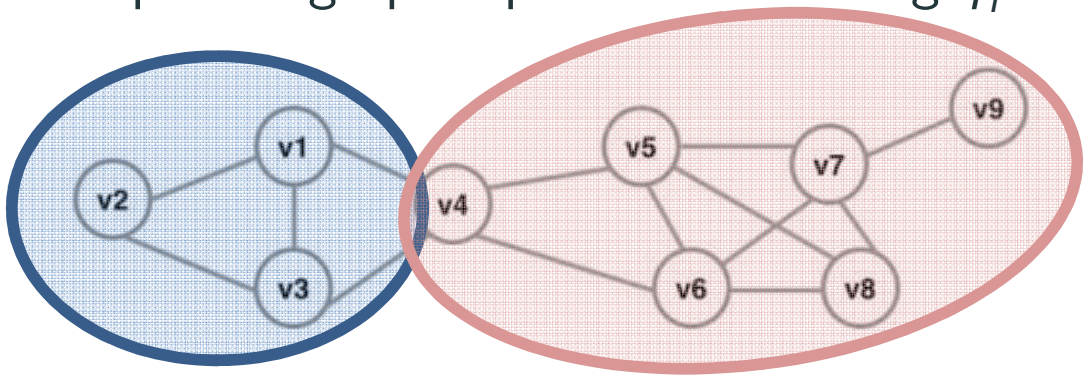
Goal: minimize the number of cut edges

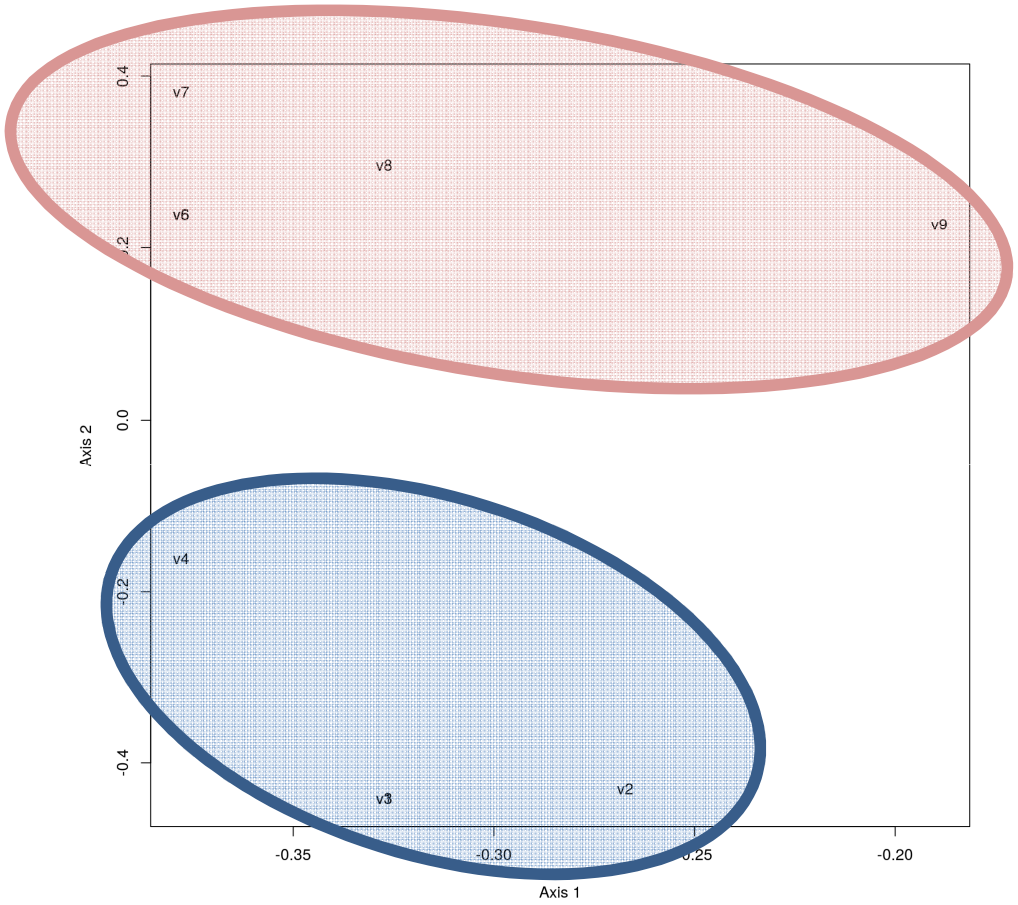
Spectral Partitionning Algorithm

1. Create L(G)
2. Compute $(\lambda_1, \vec{q_1})$ eigen element of $L(G)$
3. Choose

$$x(i) \leftarrow sign(\vec{q}^1(i))$$

Spectral graph representation using $\vec{q}_i$ as coordinates





(normalized Laplacian)