

# An introduction to graph analysis and modeling

## Descriptive Analysis of Network Data

MSc in Statistics for Smart Data – ENSAI

Autumn semester, 2018

<https://github.com/jchiquet/CourseStatNetwork>



# Outline

## ① Basic notions on graphs and networks

- Definitions

- Representations

## ② Descriptive statistics

- Vertex characteristics

- Local measurements

## ③ Graph Partitioning

- Hierarchical clustering

- Spectral Clustering

# Outline

## ① Basic notions on graphs and networks

Definitions

Representations

## ② Descriptive statistics

## ③ Graph Partitioning

# References



Statistical Analysis of Network Data: Methods and Models,  
Eric Kolaczyk  
Chapter 2, Section 1



Analyse statistique de graphes,  
Catherine Matias  
Chapitre 1

# Outline

## ① Basic notions on graphs and networks

Definitions

Representations

## ② Descriptive statistics

## ③ Graph Partitioning

# Graphs, Networks: some definitions

## Definition (Network versus Graph)

- A **Network** is a collection of interacting entities
- A **Graph** is the mathematical representation of a network

## Definition (Graph)

A graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is a mathematical structure consisting of

- a set  $\mathcal{V} = \{1, \dots, n\}$  of **vertices** or **nodes**
- a set  $\mathcal{E} = \{e_1, \dots, e_p : e_k = (i_k, j_k) \in (\mathcal{V} \times \mathcal{V})\}$  of **edges** or **links**
- The number of vertices  $N_v = |\mathcal{V}|$  is called the **order**
- The number of edges  $N_e = |\mathcal{E}|$  is called the **size**

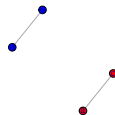
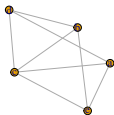
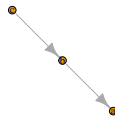
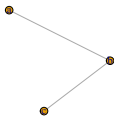
# Graphs, Networks: some vocabulary

## Not comprehensive

- subgraph  $\mathcal{H} = (\mathcal{V}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}})$ ,
- induced subgraph
- (un)directed graph,
- weighted graph,
- bipartite graph,
- tree,
- DAG, etc.

# Examples

Undirected, directed (digraph), complete, bipartite





# Paths, Cycles, Connected Components I

## Definition (Path)

In a undirected graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  a path between  $i, j \in \mathcal{V}^2$  is a series of edges  $e_1, \dots, e_k$  such that

- $\forall 1 \leq \ell < k$ , all edges  $(e_\ell, e_{\ell+1})$  share a vertex in  $\mathcal{V}$
- $e_1$  starts from  $i$ ,  $e_k$  ends to  $j$ .

## Vocabulary

- A **cycle** is a path from  $i$  to itself.
- A **connected component** is a subset  $\mathcal{V}' \subset \mathcal{V}$  such that there exists an path between any  $i, j \in \mathcal{V}'$ .
- A graph is **connected** when there is a path between every pairs of vertices.

# Paths, Cycles, Connected Components II

## Proposition (Decomposition)

*Any graph can be decomposed in a unique set of maximal connected components. The number of connected component is a least  $n - |\mathcal{E}|$*

# Neighborhood, Degree

## Definition (Neighborhood)

The neighbors of a vertex are the nodes directly connected to this vertex:

$$\mathcal{N}(i) = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}\}.$$

## Definition (Degree)

The degree  $d_i$  of a node  $i$  is given by its number of neighbors, i.e.  $|\mathcal{N}(i)|$ .

## Remark

In digraphs, vertex degree is replaced by **in-degree** and **out-degree**.

## Proposition

*In a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  the sum of the degree is given by  $2|\mathcal{E}|$ . Hence **this is always an even quantity**.*

# Outline

## ① Basic notions on graphs and networks

Definitions

Representations

## ② Descriptive statistics

## ③ Graph Partitioning

# Adjacency matrix and list of edges

## Definition (Adjacency matrix)

The connectivity of  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is captured by the  $|\mathcal{V}| \times |\mathcal{V}|$  matrix  $\mathbf{A}$ :

$$(\mathbf{A})_{ij} = \begin{cases} 1 & \text{if } i \sim j, \\ 0 & \text{otherwise.} \end{cases}$$

## Proposition

*The degree of  $\mathcal{G}$  are then simply obtained as the row-wise and/or column-wise sums of  $\mathbf{A}$ .*

## Remark

If the list of vertices is known, the only information which needs to be stored is the list of edges. In terms of storage, this is equivalent to a sparse matrix representation.

# Incidence matrix

## Definition (Incidence matrix)

The connectivity of  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is captured by the  $|\mathcal{V}| \times |\mathcal{E}|$  matrix  $\mathbf{B}$ :

$$(\mathbf{B})_{ij} = \begin{cases} 1 & \text{if } i \text{ is incident to edge } j, \\ 0 & \text{otherwise.} \end{cases}$$

## Proposition (Relationship)

Let  $\tilde{\mathbf{B}}$  be a modified *signed* version of  $\mathbf{B}$  where  $\tilde{B}_{ij} = 1 / -1$  if  $i$  is incident to  $j$  as tail/head. Then

$$\tilde{\mathbf{B}}\tilde{\mathbf{B}}^T = \mathbf{D} - \mathbf{A},$$

where  $\mathbf{D} = \text{diag}(\{d_i, i \in \mathcal{V}\})$  is the diagonal matrix of degrees.

$\rightsquigarrow \tilde{\mathbf{B}}\tilde{\mathbf{B}}^T$  is called the Laplacian matrix and will be studied latter.

# Layout and Visualization

- Visualization of large networks is a field of research in its own
- Be carefull with graphical interpretation of (large) networks

```
library(igraph)
library(sand)
GLattice <- graph.lattice(c(5,5,5))
GBlog <- aidsblog
```

# Layout and Visualization

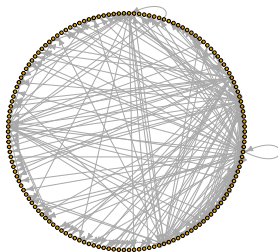
Example with circle plot

```
par(mfrow=c(1,2))  
plot(GLattice, layout=layout.circle); title("5x5x5 lattice")  
plot(GBlog , layout=layout.circle); title("blog network")
```

5x5x5 lattice



blog network



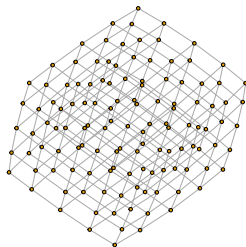


# Layout and Vizualization

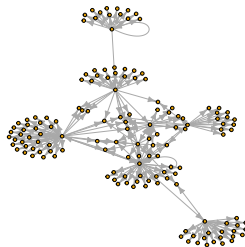
Example with Fruchterman and Reingold

```
par(mfrow=c(1,2))  
plot(GLattice, layout=layout.fruchterman.reingold); title("5x5x5 lattice")  
plot(GBlog , layout=layout.fruchterman.reingold); title("blog network")
```

5x5x5 lattice



blog network



# Layout and Visualization: **ggraph** way I

```
library(ggraph)
library(gridExtra)
g1 <- ggraph(GBlog, layout = "fr") +
  geom_edge_link(color = "lightgray") + geom_node_point() + theme_void()

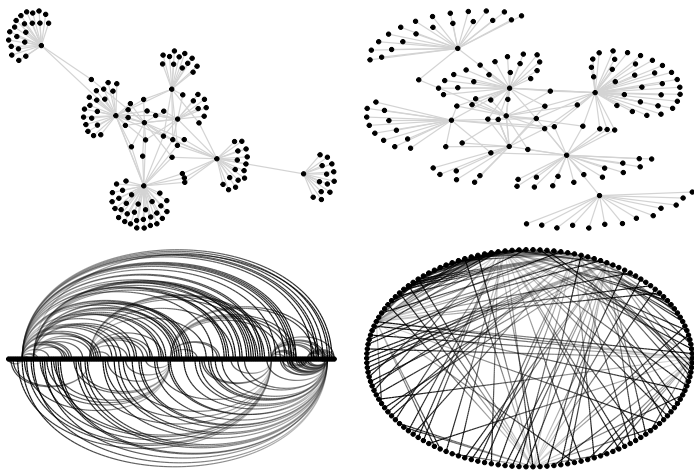
g2 <- ggraph(GBlog, layout = "kk") +
  geom_edge_link(color = "lightgray") + geom_node_point() + theme_void()

g3 <- ggraph(GBlog, layout = "linear") +
  geom_edge_arc(aes(alpha=..index..), show.legend = FALSE) +
  geom_node_point() + theme_void()

g4 <- ggraph(GBlog, layout = "linear", circular = TRUE) +
  geom_edge_link(aes(alpha=..index..), show.legend = FALSE) +
  geom_node_point() + theme_void()

grid.arrange(g1, g2, g3, g4, nrow = 2, ncol = 2)
```

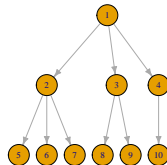
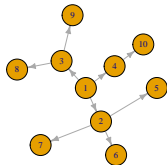
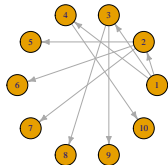
## Layout and Visualization: **ggraph** way II



# Layout and Vizualization

Do not be fooled by the plot

```
g.tree <- graph.formula(1-+2,1-+3,1-+4,2-+5,2-+6,2-+7, 3-+8,3-+9,4-+10)
par(mfrow=c(1, 3))
igraph.options(vertex.size=30, edge.arrow.size=0.5, vertex.label=NULL)
plot(g.tree, layout=layout.circle)
plot(g.tree, layout=layout.reingold.tilford(g.tree, circular=T))
plot(g.tree, layout=layout.reingold.tilford)
```



# Outline

① Basic notions on graphs and networks




② Descriptive statistics

Vertex characteristics

Local measurements

③ Graph Partitioning

# References

-  Statistical Analysis of Network Data: Methods and Models,  
Eric Kolaczyk  
Chapter 4, Sections 2 and 3
-  Statistical Analysis of Network Data with R,  
Eric Kolaczyk, Gábor Csárdi  
Chapter 4, Sections 2 and 3
-  Analyse statistique de graphes,  
Catherine Matias  
Chapitre 2

# Outline

- 1 Basic notions on graphs and networks
- 2 Descriptive statistics
  - Vertex characteristics
  - Local measurements
- 3 Graph Partitioning

# Vertex degree

## Definition (Degree distribution)

In a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , recall that  $d_i$  counts the number of incident edges in  $\mathcal{E}$  to  $i$ . Define  $f_d$  to be the fraction of vertices  $i \in \mathcal{V}$  with degree  $d_i = d$ . The collection  $\{f_d, d \geq 0\}$  is called the **degree distribution** of  $\mathcal{G}$ .

## Property

Many real world networks have a degree distribution fitting well power law distributions:

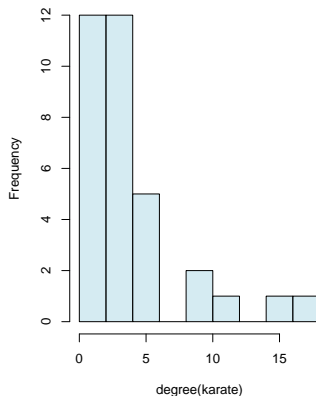
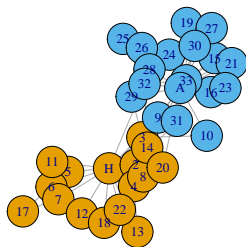
$$f_{d_i}(d) = \mathbb{P}(d_i = d) = \frac{c}{d^\gamma}, \quad c \in \mathbb{R}, \gamma > 0.$$

Those heavy-tail distributions describe few vertices with very high degrees.



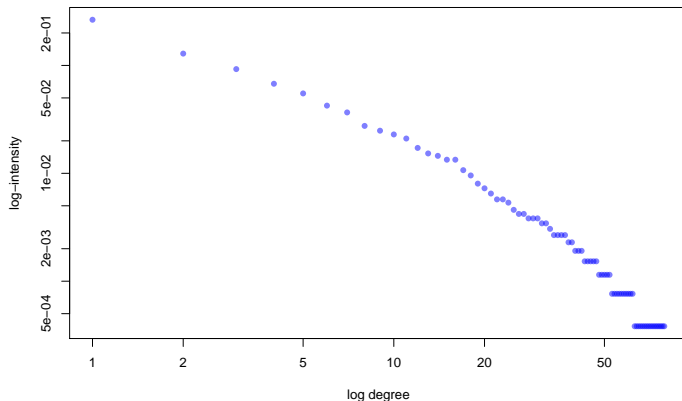
# Vertex degree: example I

```
library(sand) data(karate)
par(mfrow=c(1,2)) plot(karate)
hist(degree(karate), col=adjustcolor("lightblue", alpha.f = 0.5), main="")
```



# Vertex degree: example II

```
library(igraphdata) data(yeast)
degrees.yeast <- rev(sort(degree.distribution(yeast)))
plot(degrees.yeast[degrees.yeast!=0], log="xy", col=adjustcolor("blue", alpha.f =
0.5), pch=16, xlab="log degree", ylab="log-intensity")
```



# Joint vertex degree distribution

Definition (Empirical distribution of  $(d_i, d_j)$ )

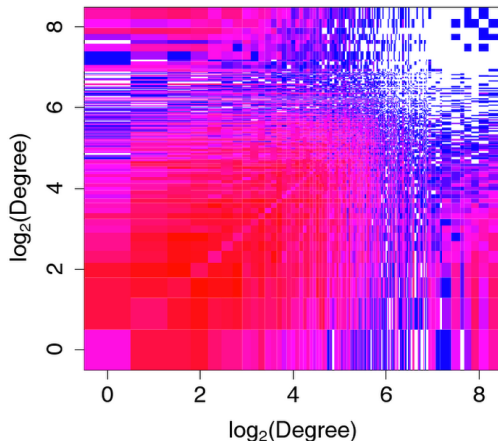
Let  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  be an undirected network and let  $N(k, \ell)$  be the number of edges whose nodes have respective degrees equal to  $(k, \ell)$  or  $(\ell, k)$ . Then the empirical distribution of  $(d_i, d_j)$  is given by

$$f_{k\ell} = \begin{cases} N(k, \ell)/2|\mathcal{E}| & \text{if } k < \ell \\ N(\ell, k)/2|\mathcal{E}| & \text{if } k > \ell \\ N(k, k)/|\mathcal{E}| & \text{if } k = \ell \end{cases}$$

Idea/principle: What kind of nodes share an edge ?

E.g. are nodes with high degrees connected with themselves or with low degree vertices?

# Joint degree distribution: example for yeast PPI network



**Figure:** Image representation of the logarithmically transformed joint degree distribution  $\log_2 f_{kl}$  of the yeast PPI network (*source*: E. Kolaczyk)

# Distance and diameter I

## Definition (distance)

- **The Length** of a path  $e_1, \dots, e_k$  is the number of edges enterin the path (here  $k$ ).
- If two nodes  $i, j$  are connected in  $G$ , then **the distance**  $\ell_{ij}$  is the length of the shortest path between  $i$  and  $j$ . If the two nodes are not connected then  $\ell_{ij} = \infty$ .

# Distance and diameter II

## Definition (mean distance)

Mean distance in  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is defined by

$$\bar{\ell} = \frac{1}{n(n-1)} \sum_{(i,j) \in \mathcal{V}^2} \ell_{ij} = \frac{2}{n(n-1)} \sum_{i < j} \ell_{ij}.$$

## Definition (diameter)

The diameter of  $\mathcal{G}$  is the greatest distance between two nodes:

$$\text{diameter}(\mathcal{G}) = \max_{(i,j) \in \mathcal{V} \times \mathcal{V}} (\ell_{ij})$$

# Distance, Diameter: example I

```
library(Matrix)
data(ppi.CC)
diameter(ppi.CC)

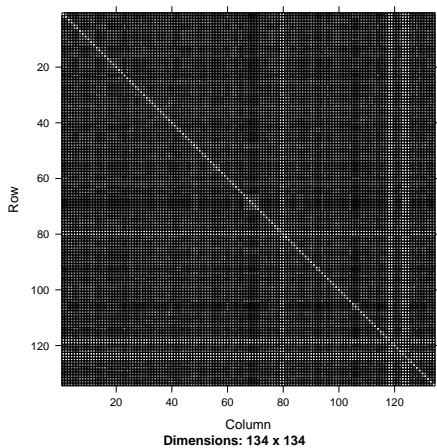
## [1] 12

average.path.length(ppi.CC)

## [1] 4.448039

image(Matrix(distances(ppi.CC)))
```

## Distance, Diameter: example II





# Vertex centrality: closeness

## Question

How important is the node/vertice in the network?

## Definition (Closeness)

Closeness is the sum of the length of the shortest paths between the node and all other nodes in the graph. It can be defined as the reciprocal of the fariness:

$$C(x) = \frac{1}{\sum_y d(y, x)}.$$

$\rightsquigarrow$  *The more central a node is, the closer it is to all other nodes.*

# Vertex centrality: betweenness

## Question

How important is the node/vertex in the network?

## Definition (Betweenness)

For every pairs of vertices, there exists at least one shortest path between the vertices such that the number of edges that the path passes through is minimized. The betweenness centrality for each vertex is the number of these shortest paths that pass through the vertex:

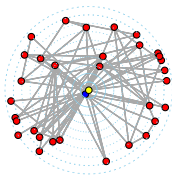
$$g(i) = \sum_{j \neq i \neq k} \frac{\sigma_{jk}(i)}{\sigma_{jk}}$$

where  $\sigma_{jk}$  is the total number of shortest paths from node  $j$  to node  $k$  and  $\sigma_{jk}(i)$  the number of those paths that pass through  $i$ .

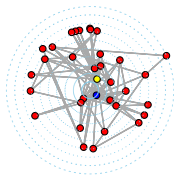
# Example for karate club data set

administrator and instructor are in blue and yellow

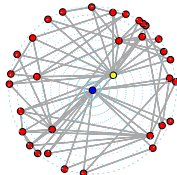
**Degree**



**Closeness**



**Betweenness**



# Jaccard Coefficient

## Definition (Jaccard Coefficient or Jaccard Index)

The Jaccard coefficient **measures similarity between finite sample sets**, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

## Example

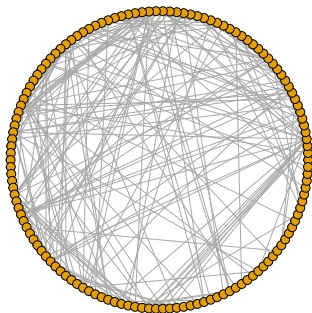
It can be used to compared two sets of egdes. For instance

- for two networks  $\mathcal{G}$  and  $\mathcal{H}$  defined on the same set of node, we can compare the sets  $\mathcal{E}_{\mathcal{G}}$  and  $\mathcal{E}_{\mathcal{H}}$ .
- for a networks  $\mathcal{G}$  we can compute similarity between nodes with the Jaccard index and use it to define a weighted graph of similarity.

# Jaccard Coefficient: example

Plot the yeast PPI interaction network

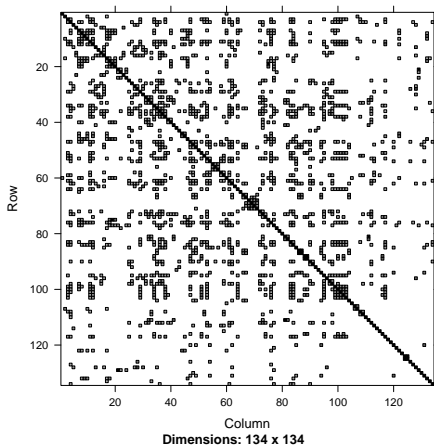
```
library(sand)
library(igraph)
plot(ppi.CC, vertex.size=6, vertex.label=NA, layout=layout_in_circle)
```



## Jaccard Coefficient: example II

Compute Jaccard similarity between vertices and give a image of this

```
library(Matrix)
image(Matrix(igraph::similarity(ppi.CC, method = "jaccard")))
```



# Outline

① Basic notions on graphs and networks

② Descriptive statistics

Vertex characteristics

Local measurements

③ Graph Partitioning

# Density

## Question

Is the network locally **dense** in some sense?

## Definition (Clique)

In an undirected graph, a clique is a subset of the vertices such that **every two distinct vertices are adjacent**.

## Definition (Density)

The density of a (sub)-graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  is defined by

$$\text{density}(\mathcal{G}) = \frac{2|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}| - 1)} = \frac{\bar{D}}{|V| - 1},$$

where  $\bar{D}$  is the mean degree of the network: how much  $\mathcal{G}$  is close to a clique?



# Clustering

## Question

Is the network locally **dense** in some sense?

## Definition (Triangle)

Triplets of vertices in the graph that are connected through a triangle. They correspond to transitive relationships. We let

- $\tau_{\Delta}(i)$  be the number of triangles in  $\mathcal{G}$  where  $i$  falls.
- $\tau_3(i)$  be the number of triplets in  $\mathcal{G}$  where  $i$  falls.

## Definition (Clustering coefficient)

$$\text{clustering}(\mathcal{G}) = \frac{1}{\mathcal{V}_2} \sum_{i \in \mathcal{V}_2} \tau_{\Delta}(i) / \tau_3(i),$$

where  $\mathcal{V}_2$  is the set of vertices whose degree is greater or equal to 2.

# Transitivity

## Question

Is the network locally **dense** in some sense?

## Definition (Triangle)

Triplet of vertices in the graph that are connected through a triangle. They correspond to transitive relationships. We let

- $\tau_{\Delta}(i)$  be the number of triangle in  $\mathcal{G}$  where  $i$  falls.
- $\tau_3(i)$  be the number of triplet in  $\mathcal{G}$  where  $i$  falls.

## Definition (Transitivity)

$$\text{transitivity}(\mathcal{G}) = \frac{\sum_{\mathcal{V}} \tau_{\Delta}(i)}{\sum_{\mathcal{V}} \tau_3(i)},$$

# Motifs

## Question

Is the network locally **dense** in some sense?

↪ This question can be answered thanks to more complicated motifs than cliques and triangles. . .

# Local density: example

Create ego graphs around teacher and instructor

```
data(karate)

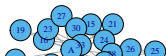
ego.instr <- induced.subgraph(karate, neighborhood(karate, 1, 1)[[1]])

## Error in FUN(X[[i]], ...): as.sociomatrix.sna input must be an adjacency
## matrix/array, network, or list.

ego.admin <- induced.subgraph(karate, neighborhood(karate, 1, 34)[[1]])

## Error in FUN(X[[i]], ...): as.sociomatrix.sna input must be an adjacency
## matrix/array, network, or list.

## Error in plot(ego.instr): object 'ego.instr' not found
## Error in plot(ego.admin): object 'ego.admin' not found
```



# Local density: example II

Maximal clique size, number of triangle

```
clique.number(karate)

## [1] 5

clique.number(ego.admin)

## Error in "igraph" %in% class(graph): object 'ego.admin' not found

length(triangles(karate))

## [1] 135

length(triangles(ego.admin))

## Error in "igraph" %in% class(graph): object 'ego.admin' not found
```

# Local density: example III

## Efficient motif counts

```
cliques(karate, min = 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

count_triangles(karate)

## [1] 18 12 11 10  2  3  3  6  5  0  2  0  1  6  1  1  1  1  1  1  1
## [24]  4  1  1  1  1  1  4  3  3 13 15
```

## Local density: example IV

Look for graph density and transitivity/clustering either globally or locally

```
graph.density(karate)

## [1] 0.1390374

graph.density(ego.instr)

## Error in "igraph" %in% class(graph): object 'ego.instr' not found

graph.density(ego.admin)

## Error in "igraph" %in% class(graph): object 'ego.admin' not found

transitivity(karate)

## [1] 0.2556818

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

## [1] 0.1500000 0.1102941
```

# Outline





① Basic notions on graphs and networks

② Descriptive statistics

③ Graph Partitioning  
    Hierarchical clustering  
    Spectral Clustering



# References

-  Statistical Analysis of Network Data: Methods and Models,  
Eric Kolaczyk  
Chapter 4, Section 4
-  Statistical Analysis of Network Data with R,  
Eric Kolaczyk, Gábor Csárdi  
Chapter 4, Section 4
-  Analyse statistique de graphes,  
Catherine Matias  
Chapitre 3
-  A Tutorial on Spectral Clustering,  
Ulrike von Luxburg

# Principle of graph partitionning

## Definition (Partition)

A decomposition  $\mathcal{C} = \{C_1, \dots, C_K\}$  of the vertices  $\mathcal{V}$  such that

- $C_k \cap C_{k'} = \emptyset$  for any  $k \neq k'$
- $\bigcup_k C_k = \mathcal{V}$

## Goal of graph partitionning

Form a partition of the vertices with unsupervised approach where the  $\mathcal{C}$  is composed by "cohesive" sets of vertices, for instance,

- ① vertices well connected among themselves
- ② well separated from the remaining vertices

# Outline

- 1 Basic notions on graphs and networks
- 2 Descriptive statistics
- 3 Graph Partitioning
  - Hierarchical clustering
  - Spectral Clustering

# Principle

**Input:**  $n$  individuals with  $p$  attributes)

1. Compute the dissimilarity between groups
2. Regroup the two most similar elements

Iterate until all element are in a single group

**Output:**  $n$  nested partitions from  $\{\{1\}, \dots, \{n\}\}$  to  $\{\{1, \dots, n\}\}$

**Algorithm 1:** Agglomerative hierarchical clustering

## Ingredients

- ① a dissimilarity measure between singleton
- ② a distance measure between sets

# Dissimilarity measures

## Standards

Use standard distances on adjacency matrix:

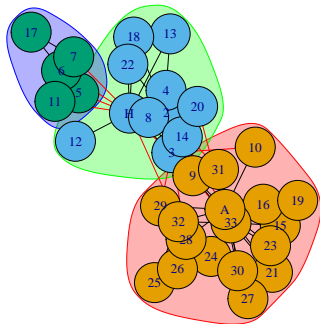
- Euclidean distance:  $x_{ij} = \sqrt{\sum_{ij} (A_{ik} - A_{jk})^2}$
- Manhattan distance:  $x_{ij} = \sum_{ij} |A_{ik} - A_{jk}|$
- etc. . .

## Graph-specific

For instance, Modularity (studied during tutorial)

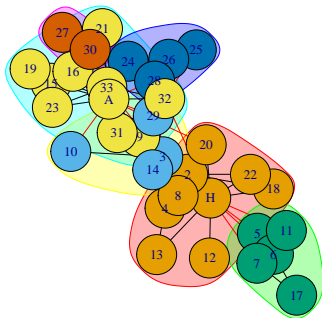
# Examples of graph clustering I

```
hc <- cluster_fast_greedy(karate)  
plot(hc, karate)
```



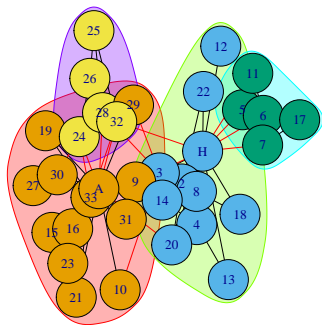
# Examples of graph clustering II

```
hc <- cluster_edge_betweenness(karate)  
plot(hc, karate)
```



# Examples of graph clustering III

```
hc <- cluster_walktrap(karate)  
plot(hc, karate)
```





# Outline

- 1 Basic notions on graphs and networks
- 2 Descriptive statistics
- 3 Graph Partitioning
  - Hierarchical clustering
  - Spectral Clustering

# Graph Laplacian

## Definition ((Un-normalized) Laplacian)

The Laplacian matrix  $\mathbf{L}$ , resulting from the modified incidence matrix  $\tilde{\mathbf{B}}$   $\tilde{B}_{ij} = 1 / -1$  if  $i$  is incident to  $j$  as tail/head, is defined by

$$\mathbf{L} = \tilde{\mathbf{B}}\tilde{\mathbf{B}}^T = \mathbf{D} - \mathbf{A},$$

where  $\mathbf{D} = \text{diag}(d_i, i \in \mathcal{V})$  is the diagonal matrix of degrees.

## Remark

- $\mathbf{L}$  is called Laplacian by analogy to the second order derivative (see below).
- Spectrum of  $\mathbf{L}$  has much to say about the structure of the graph  $\mathcal{G}$ .

# Graph Laplacian: spectrum

## Proposition (Spectrum of $\mathbf{L}$ )

*The  $n \times n$  matrix  $\mathbf{L}$  has the following properties:*

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{i,j} A_{ij} (x_i - x_j)^2, \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

- $\mathbf{L}$  is a symmetric, positive semi-definite matrix,
- the smallest eigenvalue is 0 with associated eigenvector  $\mathbf{1}$ .
- $\mathbf{L}$  has  $n$  positive eigenvalues  $0 = \lambda_1 < \dots < \lambda_n$ .

## Corollary (Spectrum and Graph)

- The multiplicity of the first eigen value (0) of  $\mathbf{L}$  determines the number of connected components in the graph.
- The larger the second non trivial eigenvalue, the higher the connectivity of  $\mathcal{G}$ .

# Normalized Graph Laplacian

## Definition ((Normalized) Laplacian)

The normalized Laplacian matrix  $\mathbf{L}$  is defined by

$$\mathbf{L}_N = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}.$$

## Proposition

*The  $n \times n$  matrix  $\mathbf{L}_N$  has the following properties:*

$$\mathbf{x}^\top \mathbf{L}_N \mathbf{x} = \frac{1}{2} \sum_{i,j} A_{ij} \left( \frac{x_i}{\sqrt{d_i}} - \frac{x_j}{\sqrt{d_j}} \right)^2, \text{quad} \forall \mathbf{x} \in \mathbb{R}^n.$$

- $\mathbf{L}$  is a symmetric, positive semi-definite matrix, with  $n$  nonnegative eigenvalues  $0 = \lambda_1 < \dots < \lambda_n$
- the smallest eigenvalue is 0 with associated eigenvector  $\mathbf{D}^{-1/2} \mathbf{1}$ .

# Absolute Graph Laplacian

## Definition ((Absolute) Laplacian)

The absolute Laplacian matrix  $\mathbf{L}_{abs}$  is defined by

$$\mathbf{L}_{abs} = \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{L}_N,$$

with eigenvalues  $1 - \lambda_n \leq \dots \leq 1 - \lambda_2 \leq 1 - \lambda_1 = 1$ , where  $0 = \lambda_1 \leq \dots \leq \lambda_n$  are the eigenvalues of  $\mathbf{L}_N$ .

# Spectral Clustering

## Principle

- ① Use the spectral property of  $\mathbf{L}$  to perform clustering in the eigen space
- ② If the network have  $K$  connected components, the first  $K$  eigenvectors are  $\mathbf{1}$  span the eigenspace associated with eigenvalue 0
- ③ Applying a simple clustering algorithm to the rows of the  $K$  first eigenvectors separate the components

↪ This principle generalizes to a graph with a single component: spectral clustering tends to separates groups of nodes which are highly connected together

# Normalized Spectral Clustering

**Input:** Adjacency matrix and number of classes  $Q$

Compute the normalized graph Laplacian  $\mathbf{L}$

Compute the eigen vectors of  $\mathbf{L}$  associated with the  $Q$  **smallest eigenvalues**

Define  $\mathbf{U}$ , the  $p \times Q$  matrix that encompasses these  $Q$  vectors

Define  $\tilde{\mathbf{U}}$ , the row-wise normalized version of  $\mathbf{U}$ :  $\tilde{u}_{ij} = \frac{u_{ij}}{\|\mathbf{U}_i\|_2}$

Apply k-means to  $(\tilde{\mathbf{U}}_i)_{i=1,\dots,p}$

**Output:** vector of classes  $\mathbf{C} \in \mathcal{Q}^p$ , such as  $C_i = q$  if  $i \in q$

**Algorithm 2:** Spectral Clustering by Ng, Jordan and Weiss (2002)

# Absolute Spectral Clustering

**Input:** Adjacency matrix and number of classes  $Q$

Compute the graph Laplacian  $\mathbf{L}_{abs}$

Compute the eigen vectors of  $\mathbf{L}_{abs}$  associated with the  $Q$  **largest** absolute eigenvalues

Define  $\mathbf{U}$ , the  $p \times Q$  matrix that encompasses these  $Q$  vectors

Apply k-means to  $(\mathbf{U}_i)_{i=1,\dots,p}$

**Output:** vector of classes  $\mathbf{C} \in \mathcal{Q}^p$ , such as  $C_i = q$  if  $i \in q$

**Algorithm 3:** Spectral Clustering by Rohe et al. (2011)



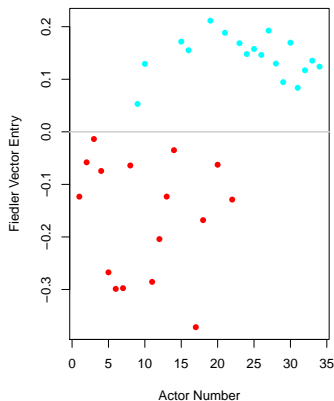
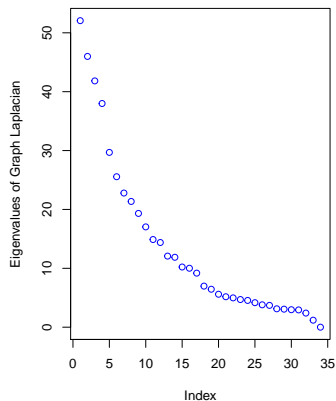
## Example: Karate club and Fielder vector and eigenvalue I

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

f.vec <- eig.anal$vectors[, 33]
faction <- igraph::get.vertex.attribute(karate, "Faction")
f.colors <- as.character(length(faction))
f.colors[faction == 1] <- "red"
f.colors[faction == 2] <- "cyan"

par(mfrow=c(1,2))
plot(eig.anal$values, col="blue",
     ylab="Eigenvalues of Graph Laplacian")
plot(f.vec, pch=16, xlab="Actor Number",
     ylab="Fiedler Vector Entry", col=f.colors)
abline(0, 0, lwd=2, col="lightgray")
```

# Example: Karate club and Fiedler vector and eigenvalue II



# Clustering based on the first non null eigenvalue

```
hc <- cluster_leading_eigen(karate)  
plot(hc, karate)
```

