

Network Analysis

Héctor Corrada Bravo

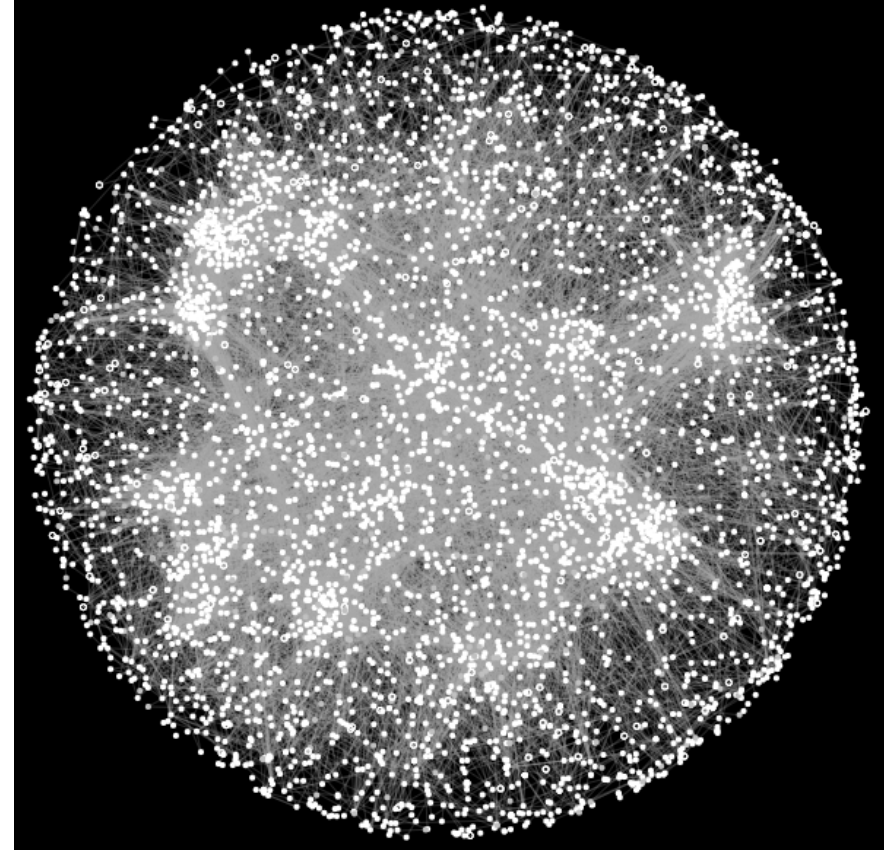
University of Maryland, College Park, USA

DATA606 2020-04-07

Genetic Interaction Network

- Yeast high-throuput double-knockdown assay
- ~5000 genes
- ~800k interactions

<http://www.geneticinteractions.org/>



Costanzo et al. (2016) Science. DOI: 10.1126/science.aaf1420

Genetic Interaction Network

- Yeast high-throuput double-knockdown assay
- ~5000 genes
- ~800k interactions

<http://www.geneticinteractions.org/>



Costanzo et al. (2016) Science. DOI: 10.1126/science.aaf1420

Genetic Interaction Network

- Number of vertices: 2803
- Number of edges: 67,268

Preliminaries

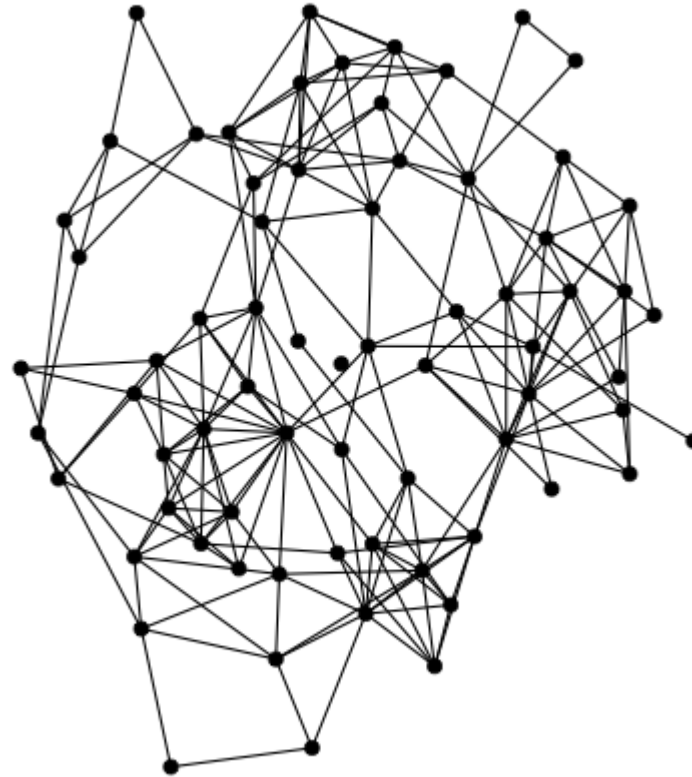
Network: abstraction of
entities and their interactions

Graph: mathematical
representation

vertices: nodes

edges: links

Undirected graph



Preliminaries

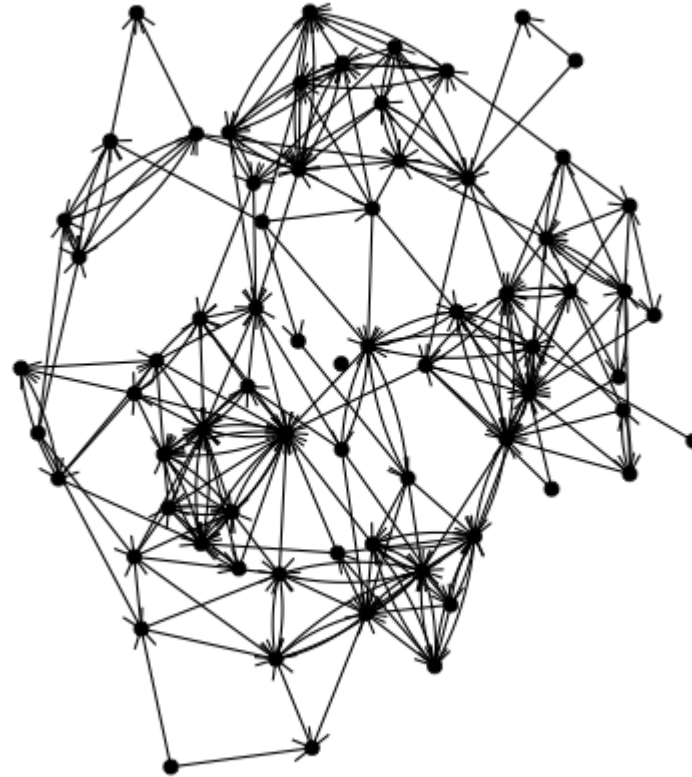
Network: abstraction of *entities* and their interactions

Graph: mathematical representation

vertices: nodes

edges: links

Directed graph



Network statistics: notation

Number of vertices: n

In our example: *number of genes*

Network statistics: notation

Number of vertices: n

In our example: *number of genes*

Number of edges: m

In our example: *number of genetic interactions*

Network statistics: notation

Number of vertices: n

In our example: *number of genes*

Number of edges: m

In our example: *number of genetic interactions*

Degree of vertex i : k_i

Number of genetic interactions for gene i

Network statistics: notation

On the board:

- Calculate number of edges m using degrees k_i (for both directed and undirected networks)
- Calculate *average degree* c
- Calculate *density* ρ

Network statistics: notation

On the board:

- Calculate number of edges m using degrees k_i (for both directed and undirected networks)
- Calculate *average degree* c
- Calculate *density* ρ

In our example:

Average degree: 47.9971459

Density: 0.0171296

(On the board)

Number of edges using degrees (undirected)

$$m = \frac{1}{2} \sum_{i=1}^n k_i$$

Number of edges using degrees (directed)

$$m = \sum_{i=1}^n k_i^{\text{in}} = \sum_{i=1}^n k_i^{\text{out}}$$

(On the board)

Average degree

$$c = \frac{1}{n} \sum_{i=1}^n k_i$$

Density

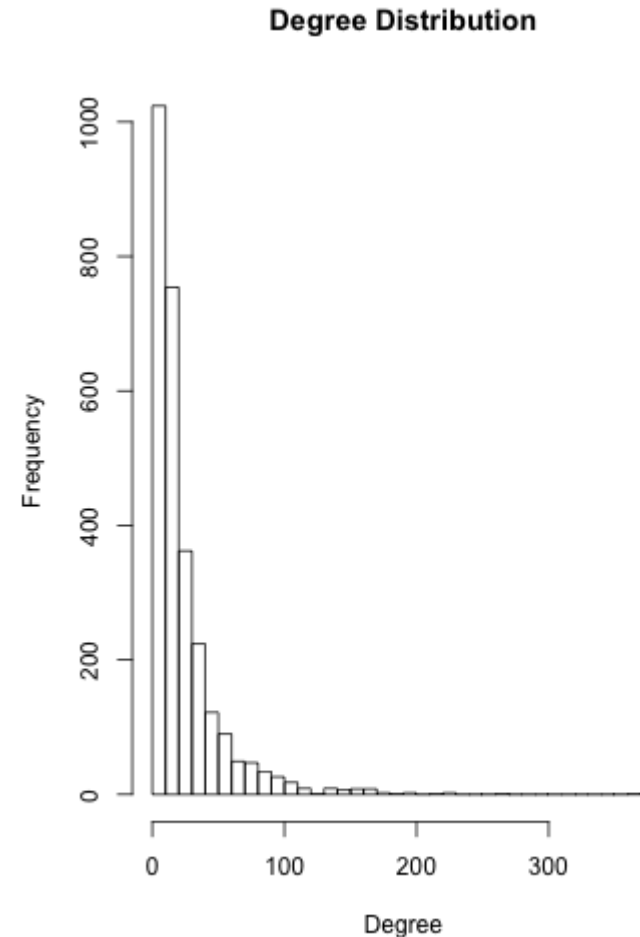
$$\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)} = \frac{c}{n-1} \approx \frac{c}{n}$$

Degree distribution

Fundamental analytical tool to characterize networks

p_k : probability randomly chosen vertex has degree k

On the board: how to calculate p_k and how to calculate average degree c using degree distribution.



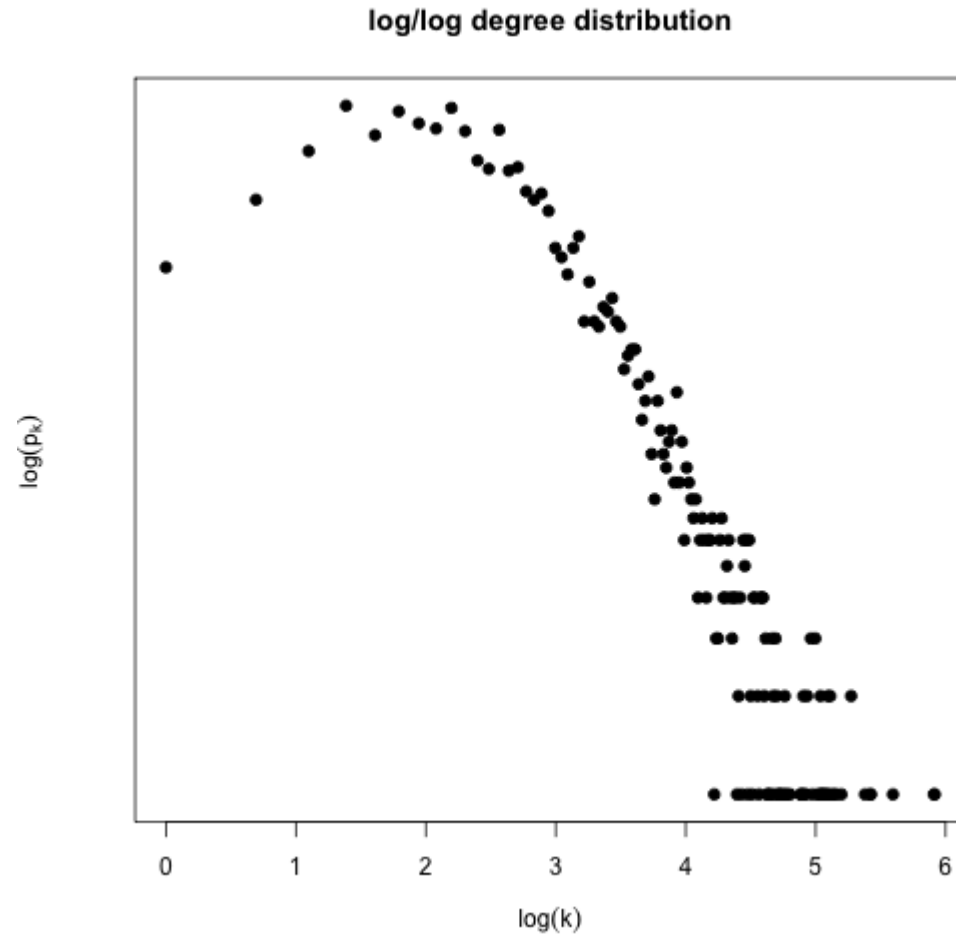
(On the board)

Degree distribution

$$p_k = \frac{n_k}{n}$$

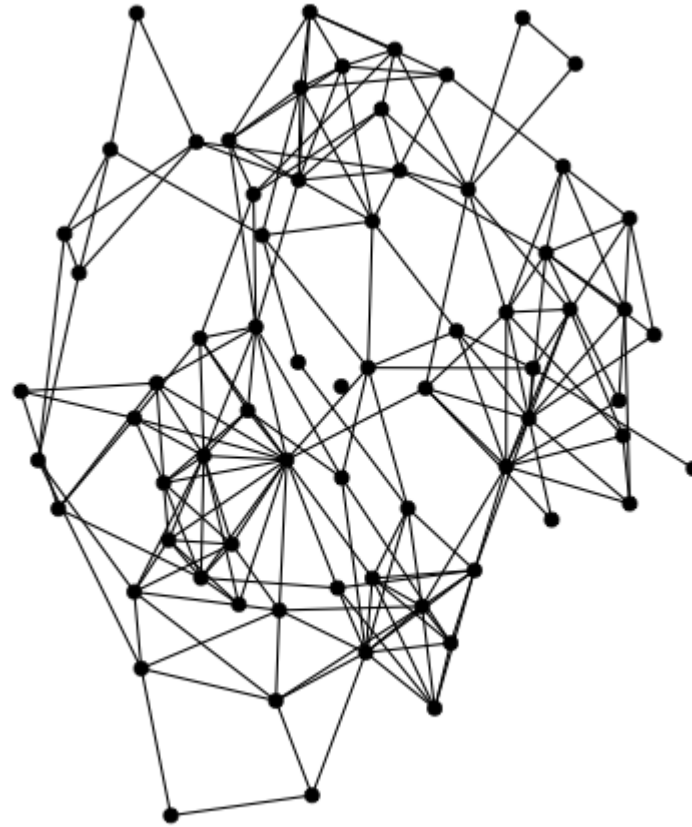
n_k : number of nodes in graph with degree k

Degree Distribution



Paths and Distances

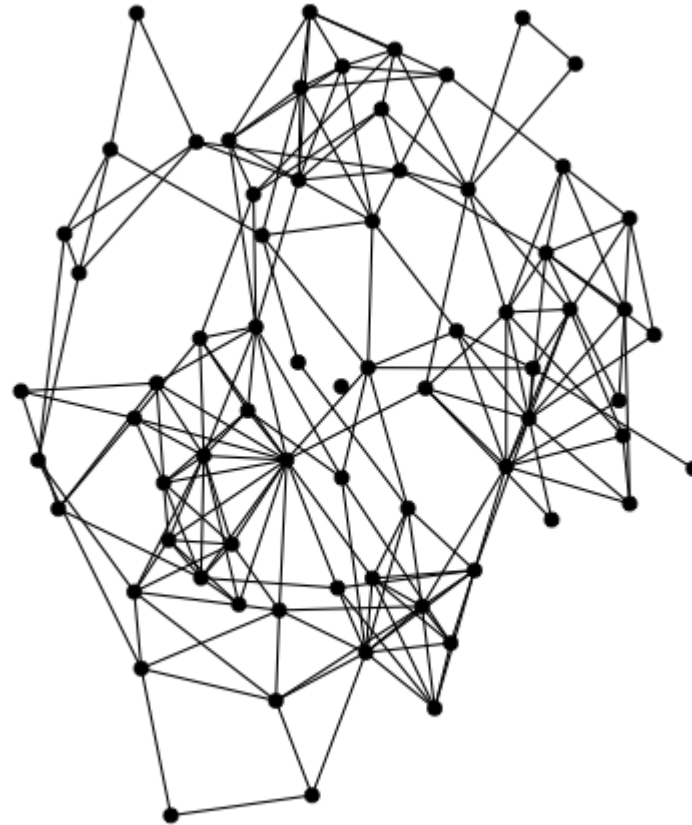
Distance d_{ij} : length of **shortest** path between vertices i and j .



Paths and Distances

Distance d_{ij} : length of **shortest** path between vertices i and j .

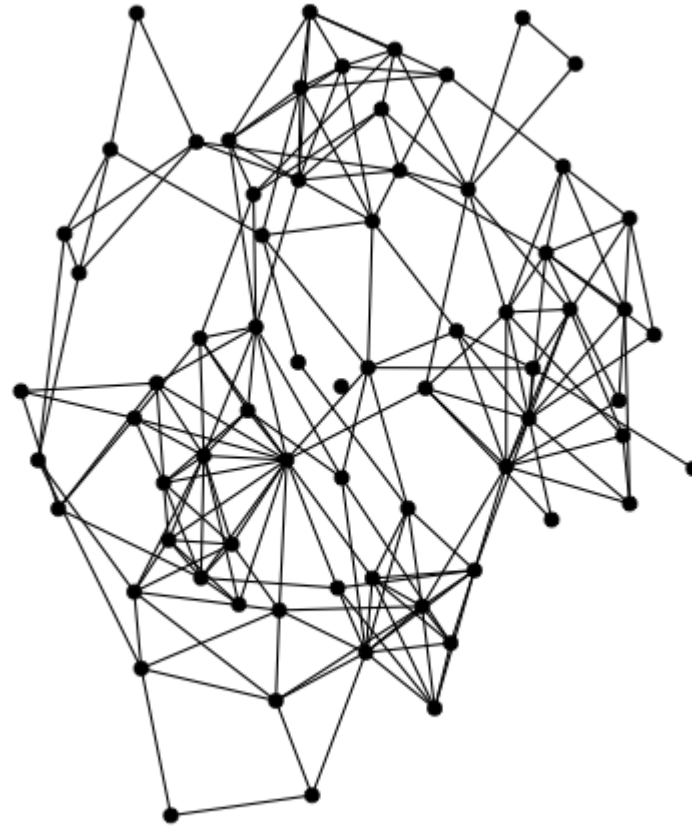
Diameter: longest shortest path $\max_{ij} d_{ij}$



Paths and Distances

Distance d_{ij} : length of **shortest** path between vertices i and j .

On the board: average path length

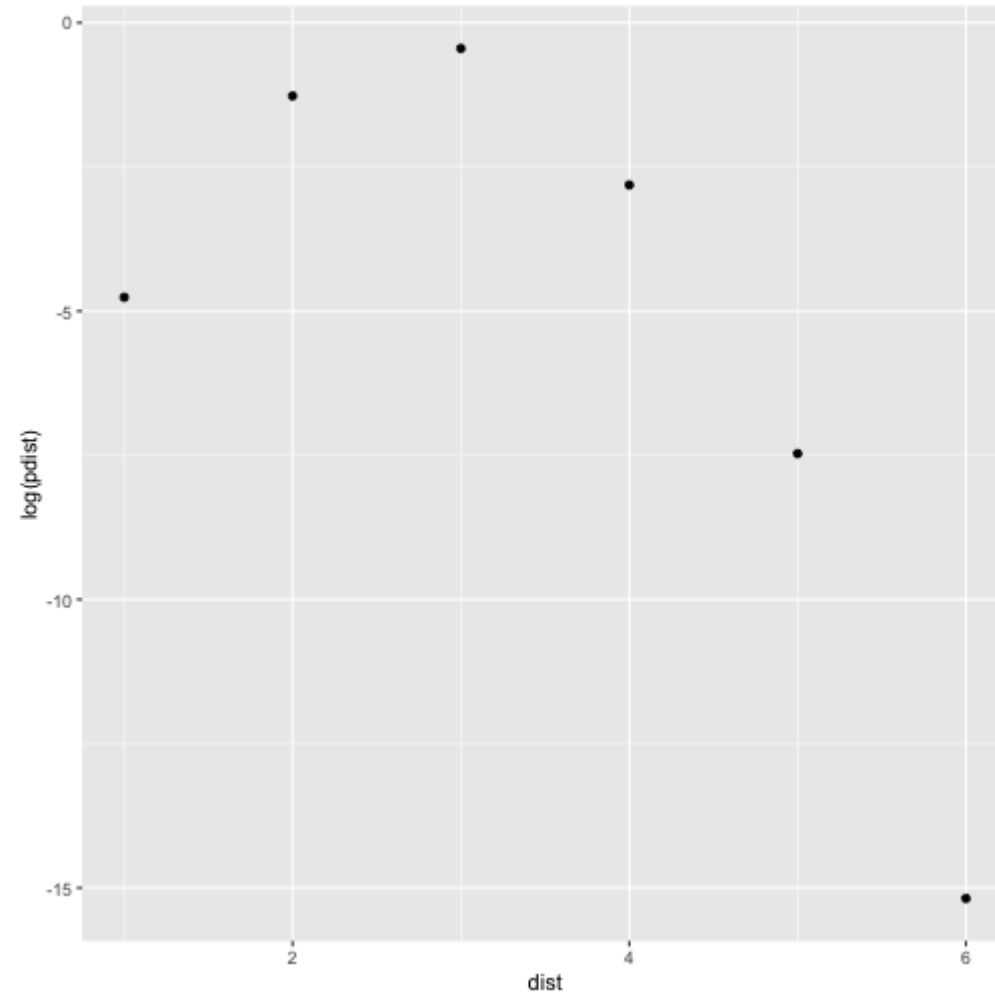


(On the board)

Average path length

$$\bar{d} = \frac{1}{n(n-1)} \sum_{i,j; i \neq j} d_{ij}$$

Distance Distribution



Distances and paths

By convention: if there is no path between vertices i and j then $d_{ij} = \infty$

Distances and paths

By convention: if there is no path between vertices i and j then $d_{ij} = \infty$

Vertices i and j are connected if $d_{ij} < \infty$

Distances and paths

By convention: if there is no path between vertices i and j then $d_{ij} = \infty$

Vertices i and j are connected if $d_{ij} < \infty$

Graph is connected if $d_{ij} < \infty$ for all i, j

Distances and paths

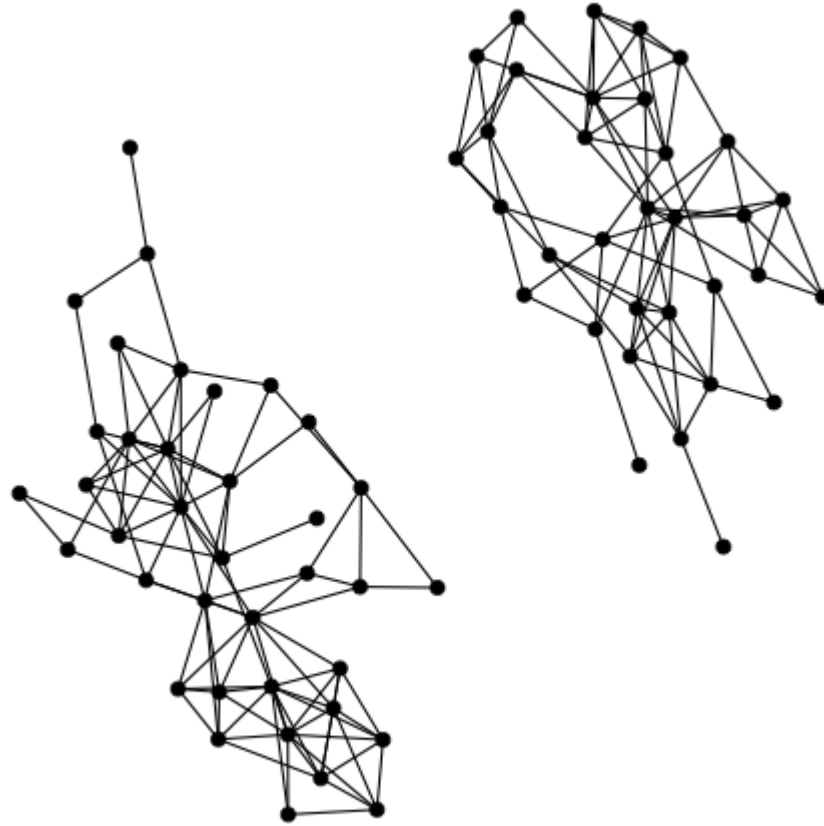
By convention: if there is no path between vertices i and j then $d_{ij} = \infty$

Vertices i and j are connected if $d_{ij} < \infty$

Graph is connected if $d_{ij} < \infty$ for all i, j

Components maximal subset of connected components

Components



Clustering Coefficient

Another quantity of interest: how dense is the neighborhood around vertex i ?

Do the genes that interact with me also interact with each other?

Related to the *locality* property.

Definition on the board

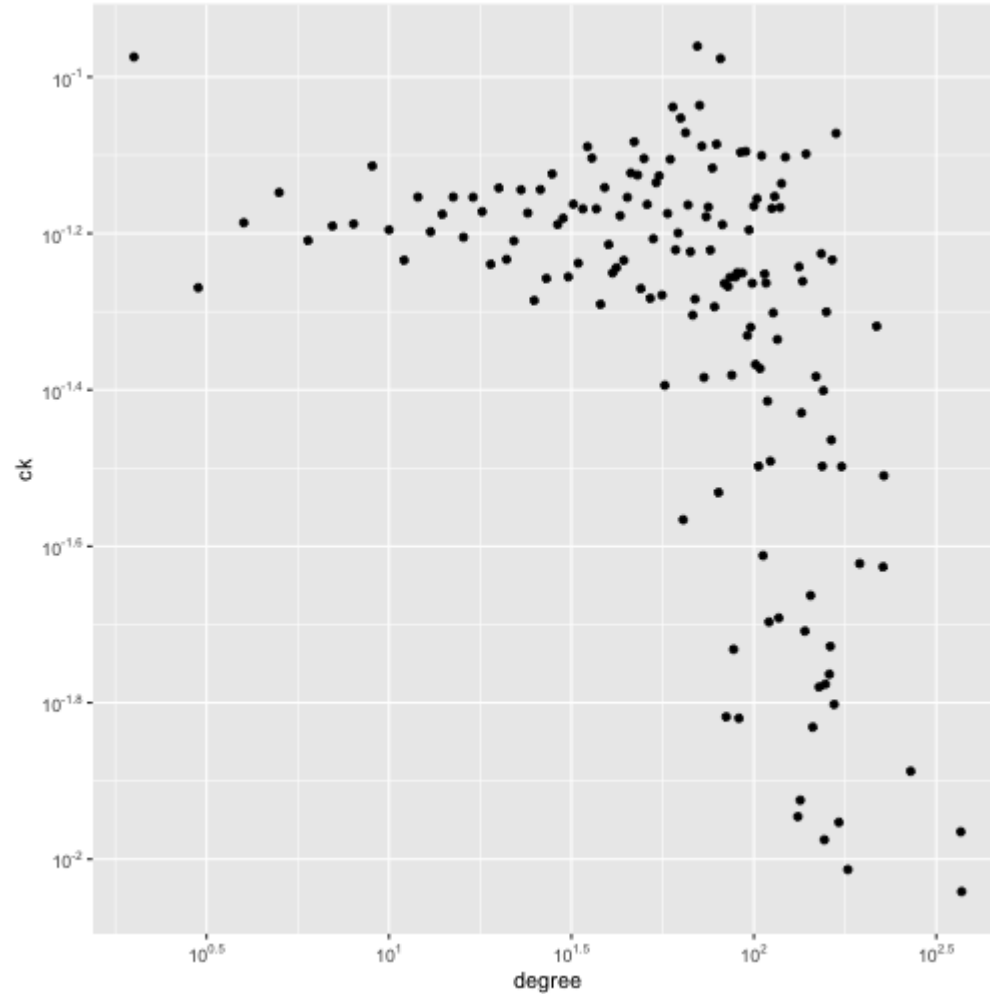
(On the board)

Clustering coefficient

$$c_i = \frac{2m_i}{k_i(k_i - 1)}$$

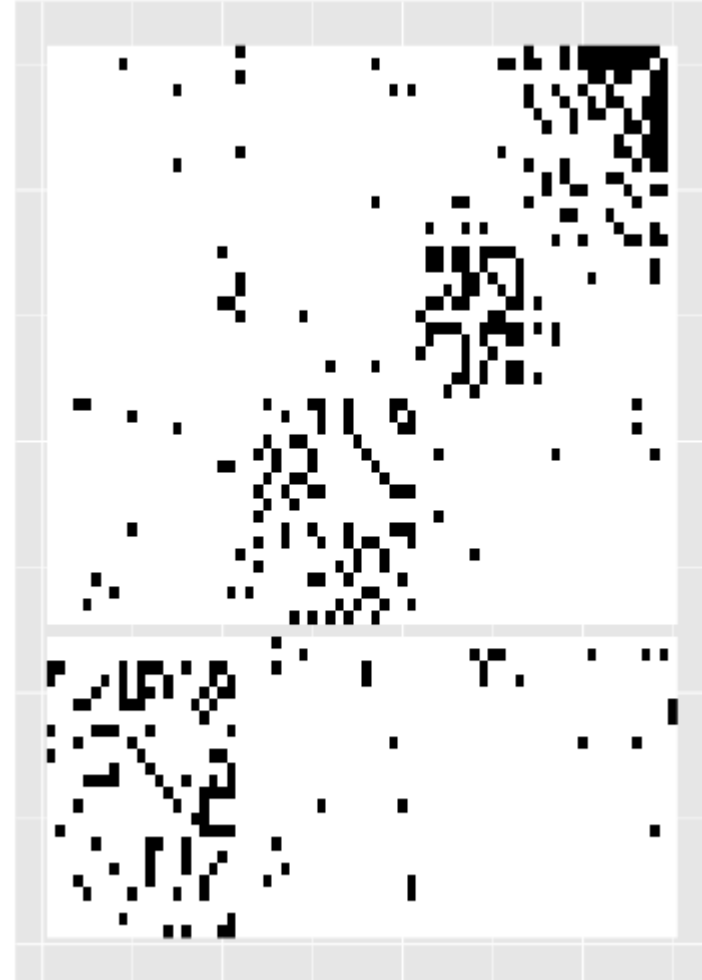
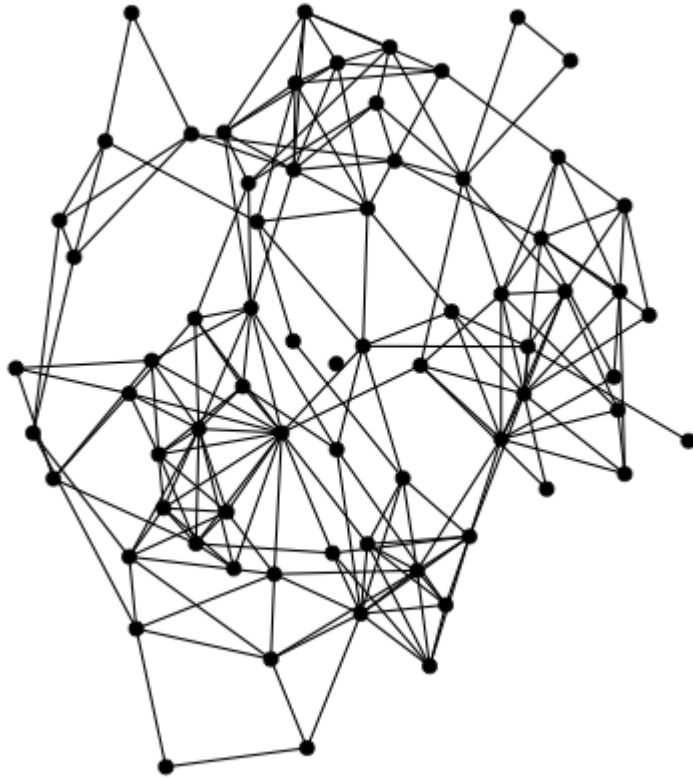
m_i : number of edges between neighbors of vertex i

Clustering coefficient



Adjacency Matrix

Undirected graph



Adjacency Matrix

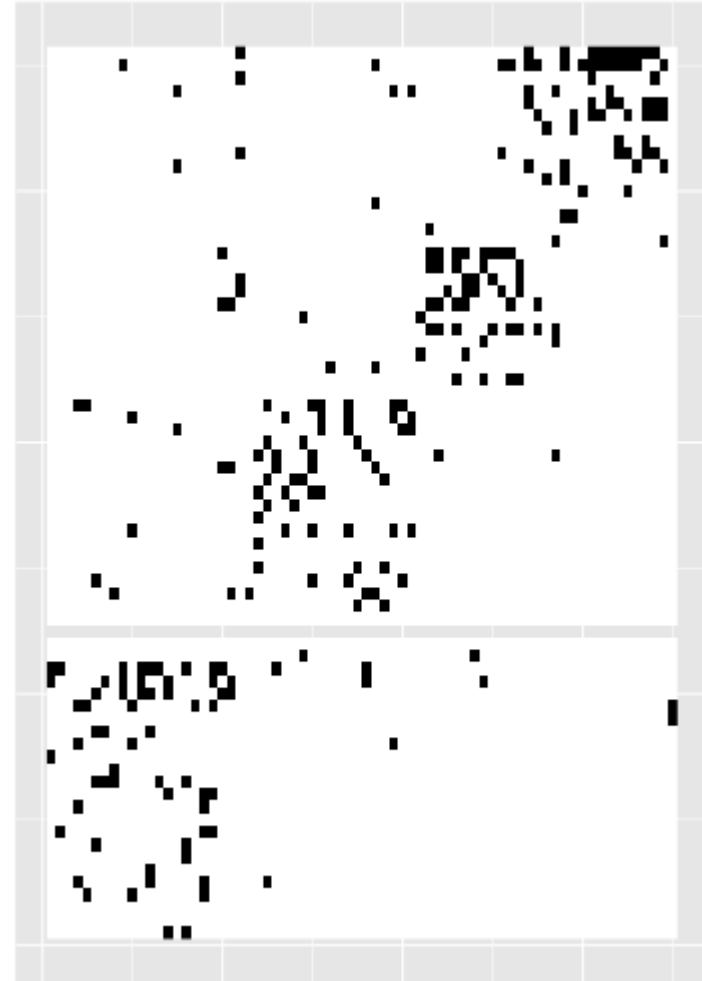
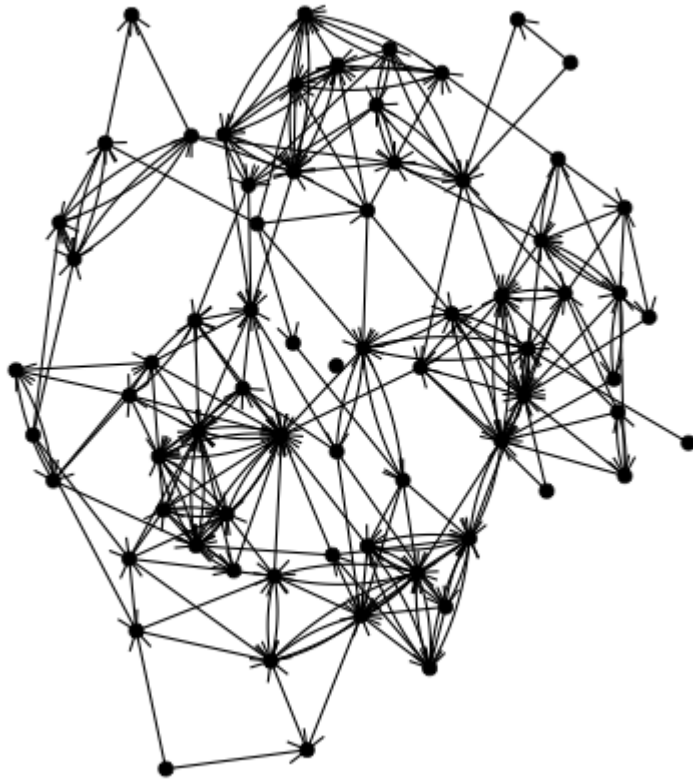
On the board:

- Definition
- Computing degree with adj. matrix
- Computing num. edges m with adj. matrix
- Computing paths with adj. matrix



Adjacency Matrix

Directed graph



Weighted networks

Edges are assigned a weight indicating quantitative property of interaction

Weighted networks

Edges are assigned a weight indicating quantitative property of interaction

- Strength of genetic interaction (evidence from experiment)
- Rates in a metabolic network
- Spatial distance in an ecological network

Weighted networks

Edges are assigned a weight indicating quantitative property of interaction

- Strength of genetic interaction (evidence from experiment)
- Rates in a metabolic network
- Spatial distance in an ecological network

Adjacency matrix contains weights instead of 0/1 entries

Weighted networks

Edges are assigned a weight indicating quantitative property of interaction

- Strength of genetic interaction (evidence from experiment)
- Rates in a metabolic network
- Spatial distance in an ecological network

Adjacency matrix contains weights instead of 0/1 entries

Path lengths are the sum of edge weights in a path

Hypergraphs

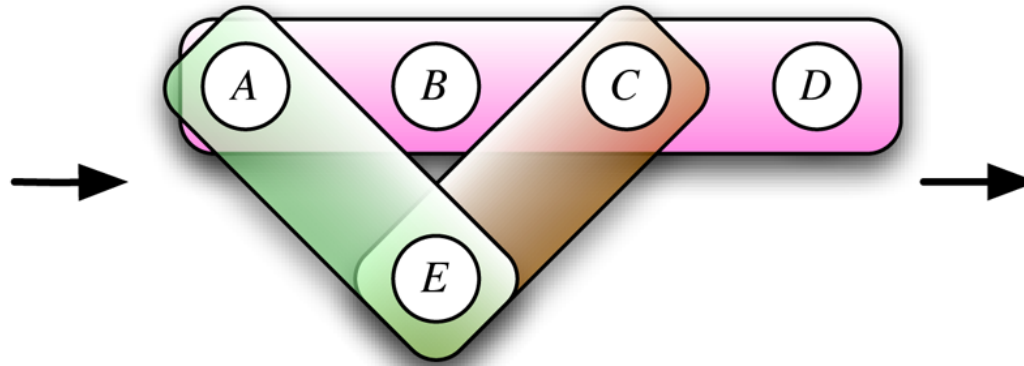
Edges connect more than two vertices

A Protein-protein interaction network

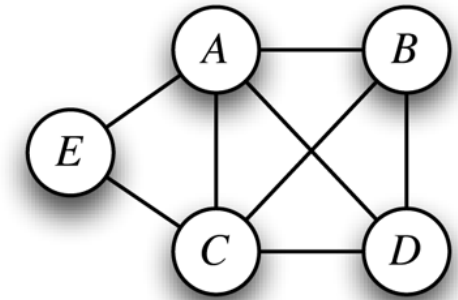
$$C_1 = \{A, B, C, D\}$$

$$C_2 = \{A, E\}$$

$$C_3 = \{C, E\}$$



Hypergraph



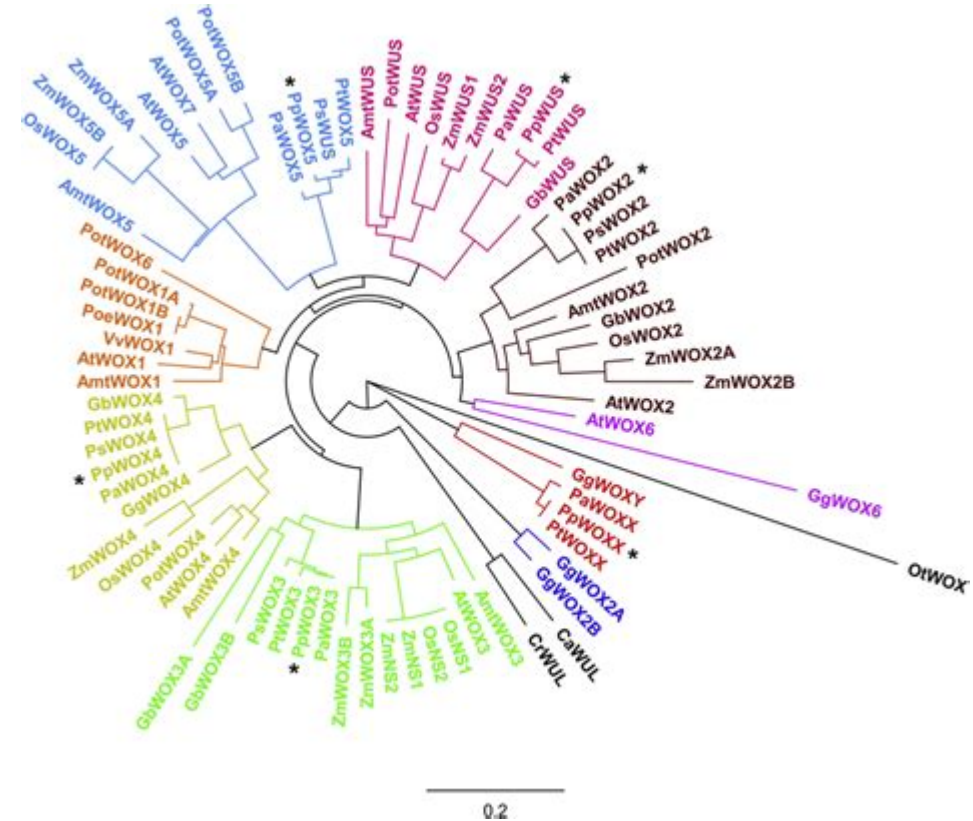
Graph

<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000385>

Trees

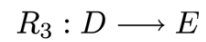
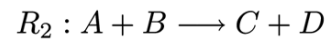
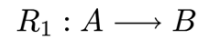
Acyclic graphs

Single path between any pair of vertices

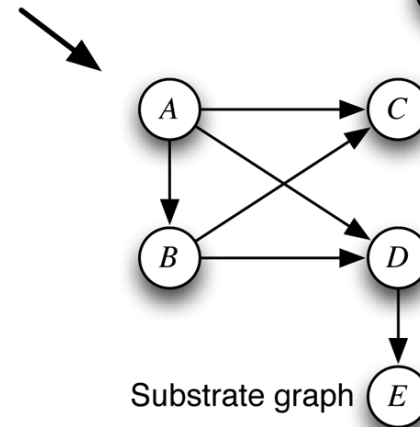
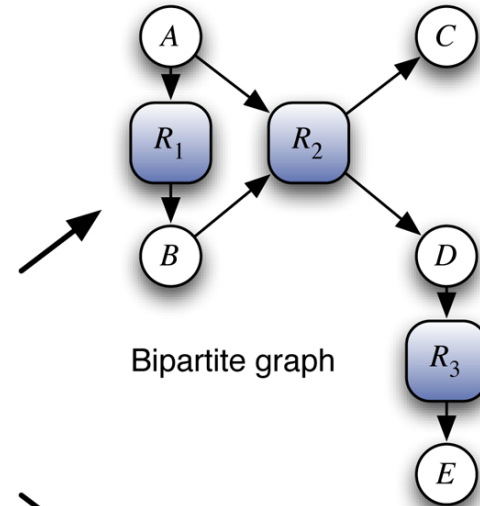
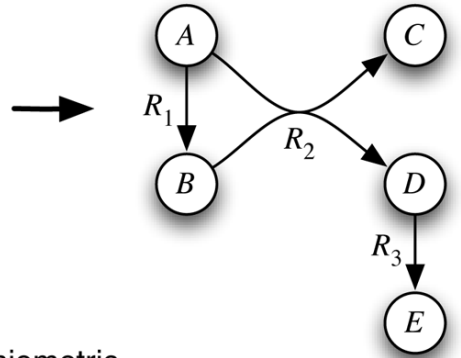


Bipartite Networks

C Reaction networks



$$\begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} \begin{pmatrix} R_1 & R_2 & R_3 \\ -1 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{array}{l} \text{Stoichiometric} \\ \text{matrix} \end{array}$$



Bipartite Networks

We use an *Incidence Matrix* B instead of *Adjacency Matrix*

(On the board): definition

Bipartite Networks

Projections

vertex projection: P_{ij} , num. of groups in which vertices i and j co-occur

group projection: P'_{ij} , num. of members groups i and j share

Bipartite Networks

Projections

vertex projection: P_{ij} , num. of groups in which vertices i and j co-occur

group projection: P'_{ij} , num. of members groups i and j share

(On the board)

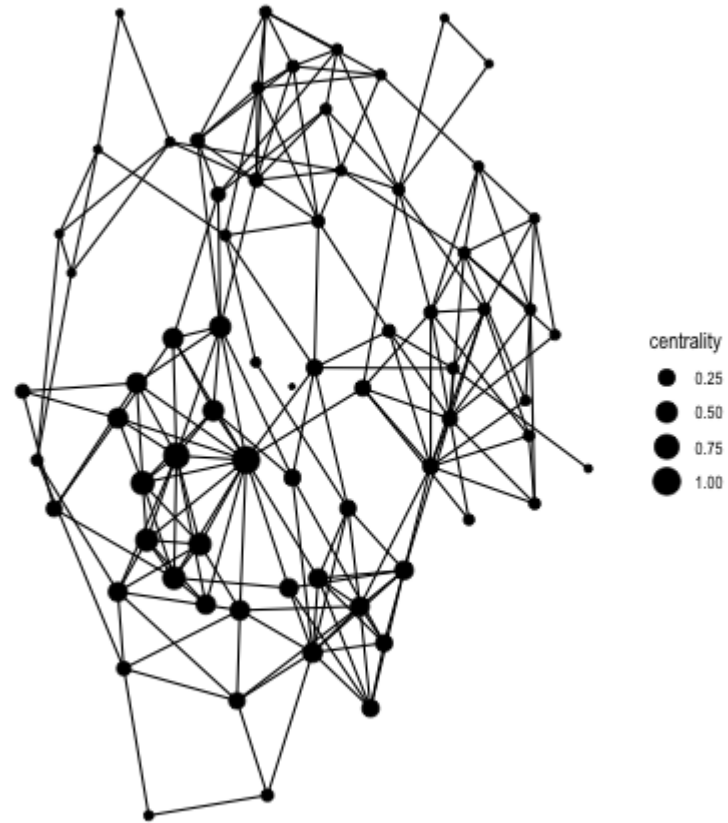
$$P = B^T B$$

$$P' = B B^T$$

Centrality

What are the *important* nodes in the network?

What are *central* nodes in the network?



Centrality

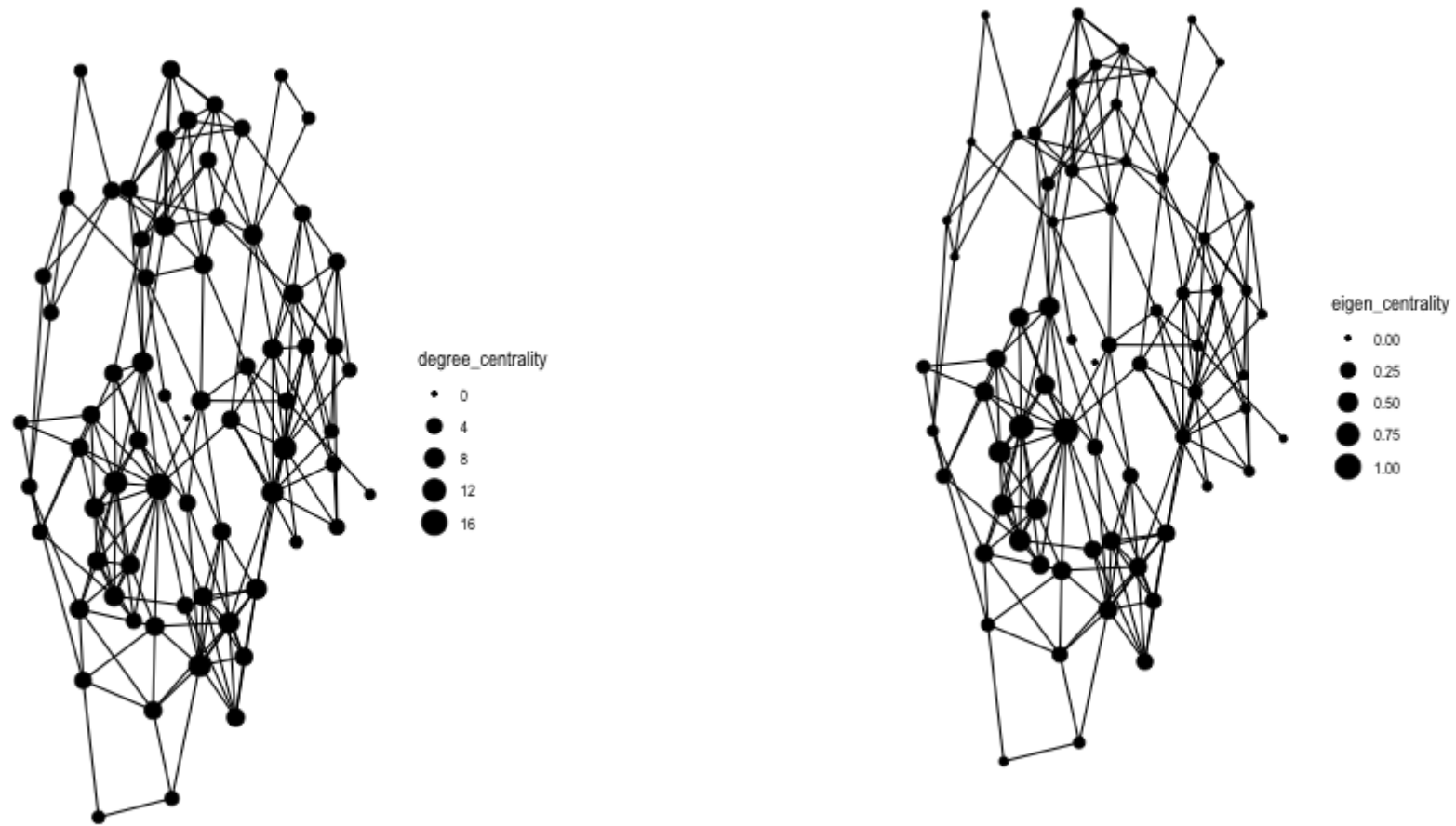
Undirected Graphs

- Eigenvalue Centrality

Directed Graphs

- Katz Centrality
- Pagerank

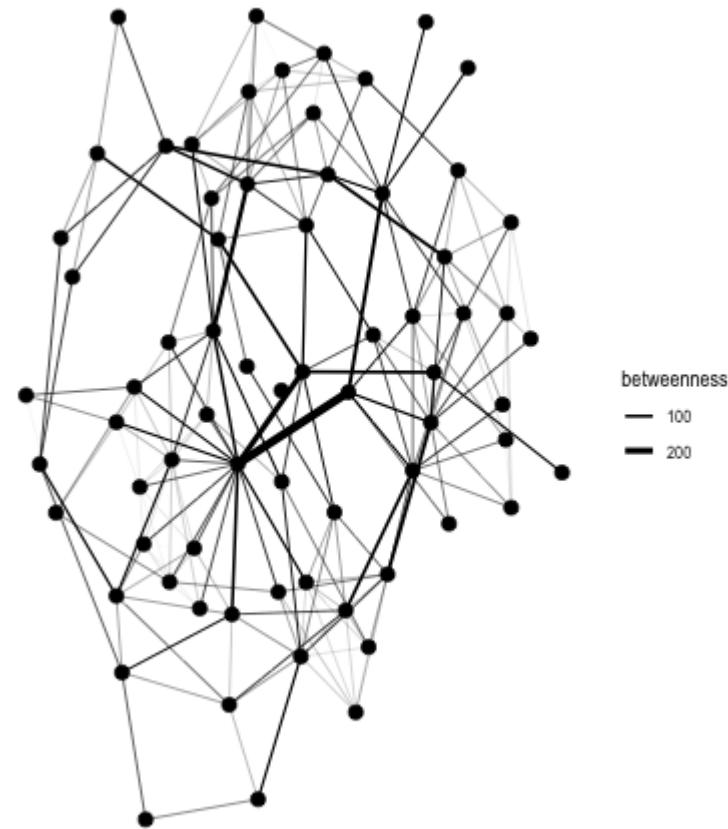
Centrality



Betweenness

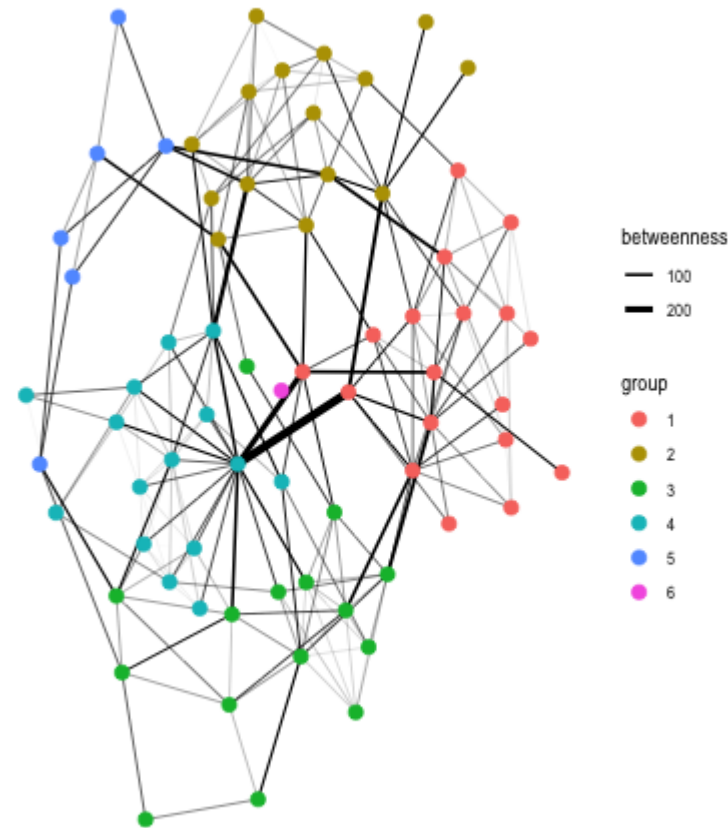
What are the *important* edges in the network?

What are edges that may connect clusters of nodes in the network?



Betweenness

Girvan-Newman Algorithm -
hierarchical method to
partition nodes into
communities using edge
betweenness



Girvan-Newman Algorithm

Two phases:

Phase One: Compute betweenness for every edge

Phase Two: Discover communities by removing *high* betweenness edges (similar to hierarchical clustering)

Girvan-Newman Algorithm

Calculating Betweenness

Formally, $\text{betweenness}(e)$: fraction of node pairs (x, y) where shortest path crosses edge e

Path Counting: For each vertex x , use breadth-first-search to count number of shortest paths through each edge e in graph between x and every other vertex y .

Sum result across vertices for each edge e , and divide by two

Sum result across nodes, and divide by two

Girvan-Newman Algorithm

Counting Paths

Algorithm (starting from node x)

1. Construct breadth-first search tree
2. (Root->Leaf) Label each vertex v with the number of shortest paths between x and v : sum of labels of parents
3. (Leaf->Root) Count the (weighted) number of shortest paths that go through each edge: next slide

Girvan-Newman Algorithm

Counting Paths

Step 3, counting number of shortest paths through each edge

a. Leafs v in search tree get a *credit* of $C_v = 1$

b. Incoming edge $e_i = (y_i, v)$ to vertex v in search tree gets *credit*

$$C_{e_i} = C_v * \frac{p_i}{\sum_j p_j}$$

- p_i : number of shortest paths between x and y_i (computed in Step 2)
- sum \sum_j is over parents of v

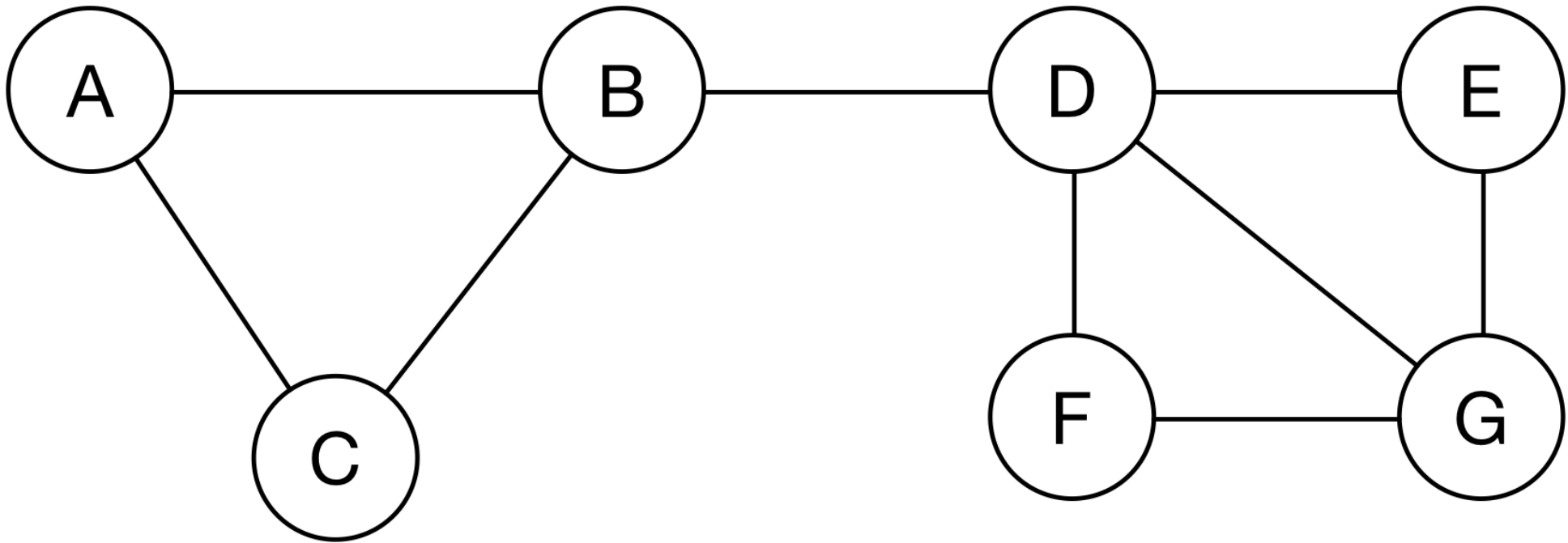
Girvan-Newman Algorithm

Counting Paths

c. Non-leaf vertex v gets credit $C_v = 1 + \sum_j e_j$ where sum j is over outgoing edges e_j in search tree

Girvan-Newman Algorithm

Example



Resources

Cross-language

igraph: <http://igraph.org/>

Resources

R

Workhorses:

- `igraph`
- `Rgraphviz`

Tidyverse (<https://tidyverse.org>):

- `tidygraph`
- `ggraph`

Resources

Python

- `igraph`
- `networkx`