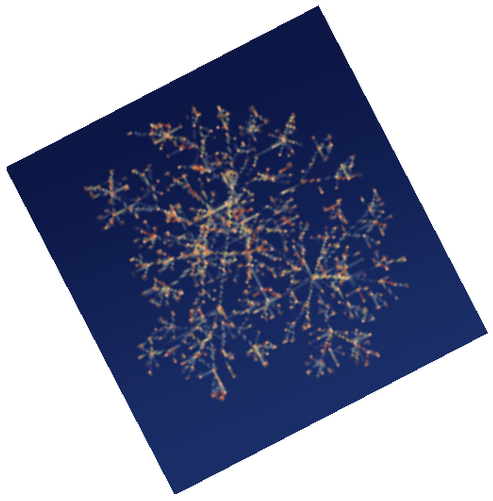
COMPLEX SYSTEMS
@CS.AALTO.FI

CS-E5740

Complex Networks

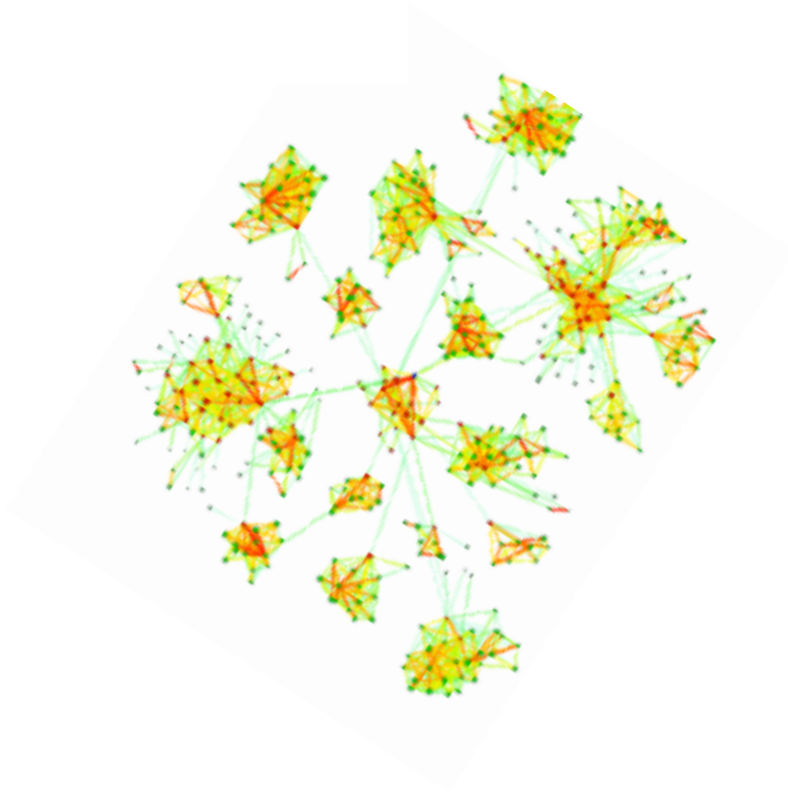


Assistants:

Tuomas Alakörkkö
Sara Heydari
Arash Badie Modiri
Ana Triana Hoyos
Enrico Glerean
Tarmo Nurmi
Javier Ureña Carrion

Lecturers:

Mikko Kivelä
mikko.kivela@aalto.fi
www.mkivela.com



fall 2018

Goals

After the course, you should

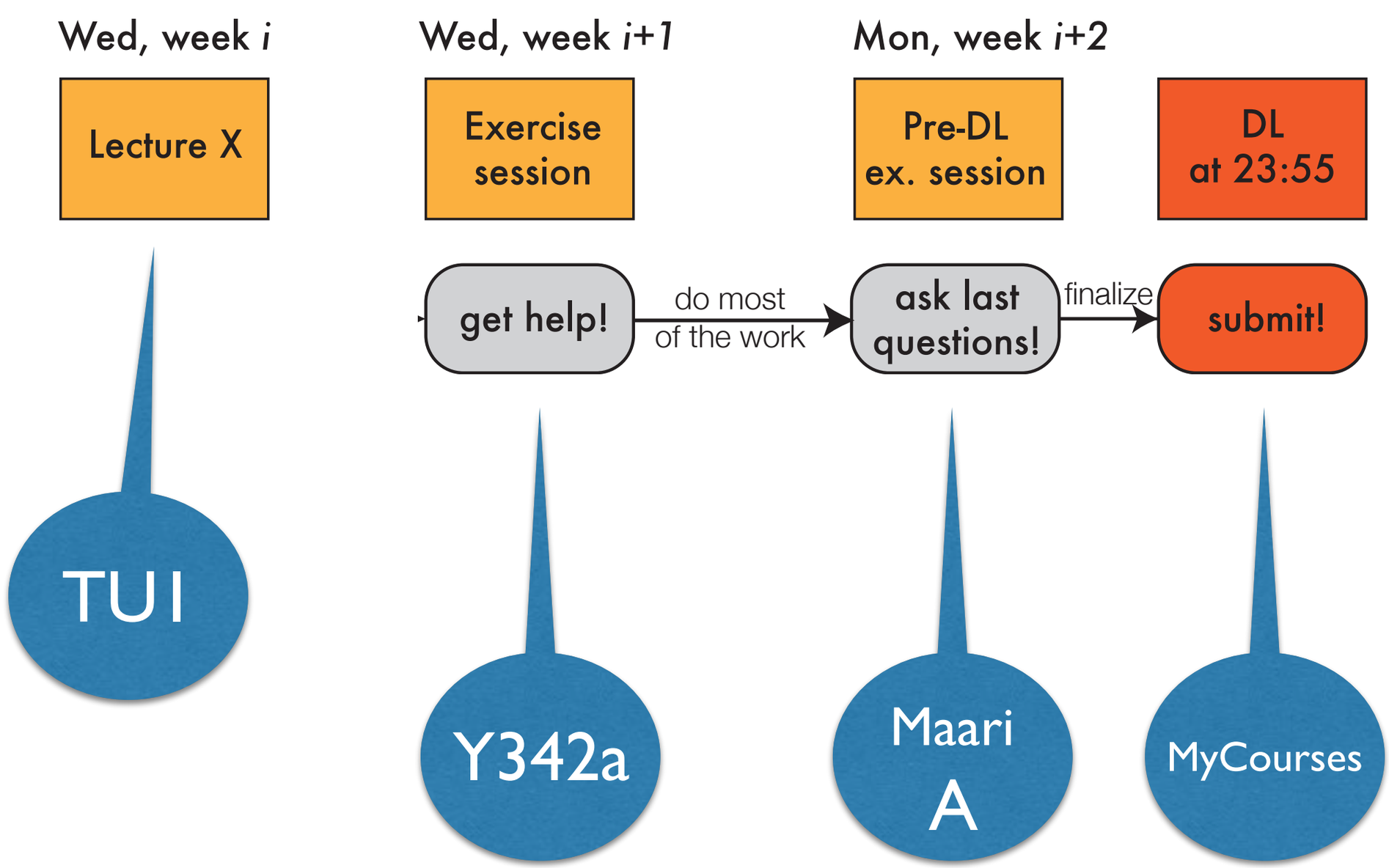
- know how to **analyze** and **characterize networks**,
- know the fundamental **network models**,
- understand how **networks evolve**, and
- know how network structure affects **dynamical processes**.

A Brief How-To

- **No exam**
- Weekly problem sets (return online)
  - 104 of 193 of overall points AND

    25 of 52 of project points needed for passing
- Project work (due 2018-12-21)
- Coming up next: tutorials
  - Python: Mon 2018-09-17 14.15 - 16.00, Maari A
  - Binning: Wed 2018-09-19 12.15 - 14.00, Y342a
- See the practicalities file in MyCourses for all the details!

Pipeline for one problem set



(Mon 29.10. -> U256)

Continuous feedback!

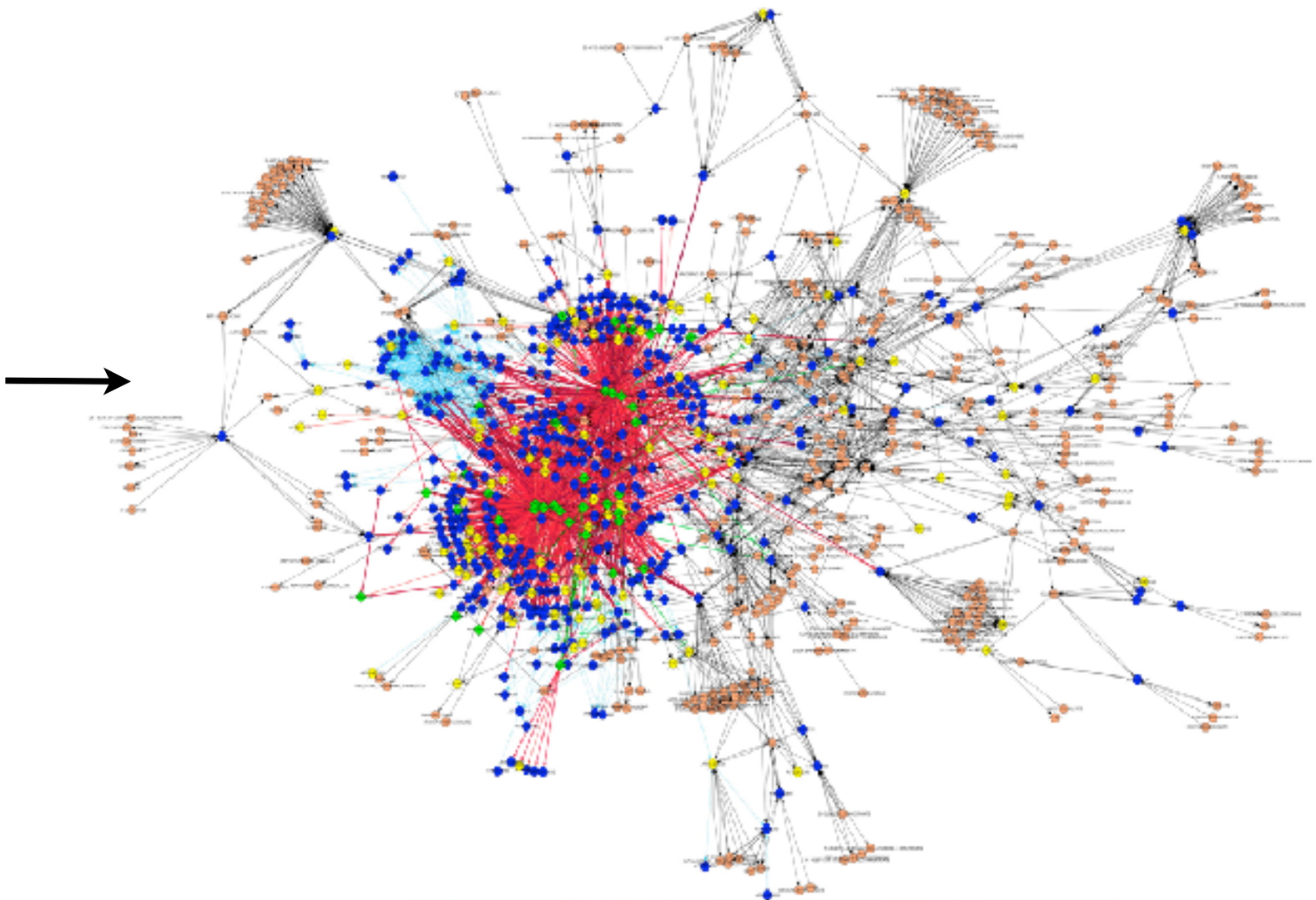
- We will collect feedback for each exercise set and the project.
- We reward you with 1 bonus point for each time you give feedback
- We will publish summaries after each round!
- General anonymous feedback also possible (see the practicalities file in MyCourses)

Course outline

1. Introduction (motivation, definitions, etc.)
2. Static network models: random and small-world networks
3. Growing network models: scale-free networks
4. Percolation, error & attack tolerance of networks, epidemic models
5. Network analysis
6. Social networks & (socio)dynamic models
7. Weighted networks
8. Clustering, sampling, inference
9. Temporal networks & multilayer networks

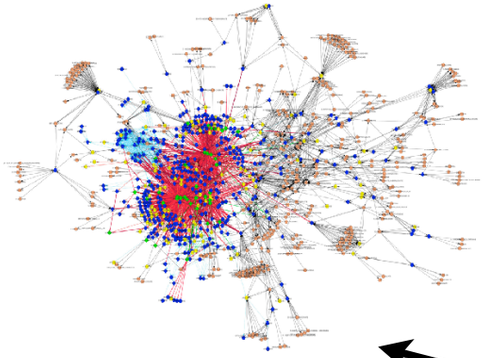
Part I:
Why Study Networks?

Everything is a network

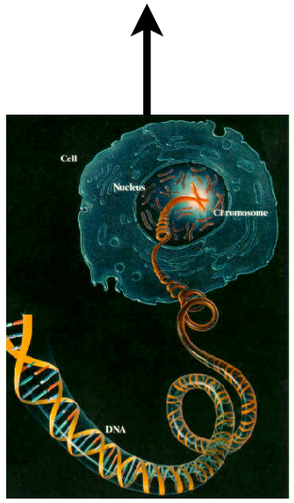


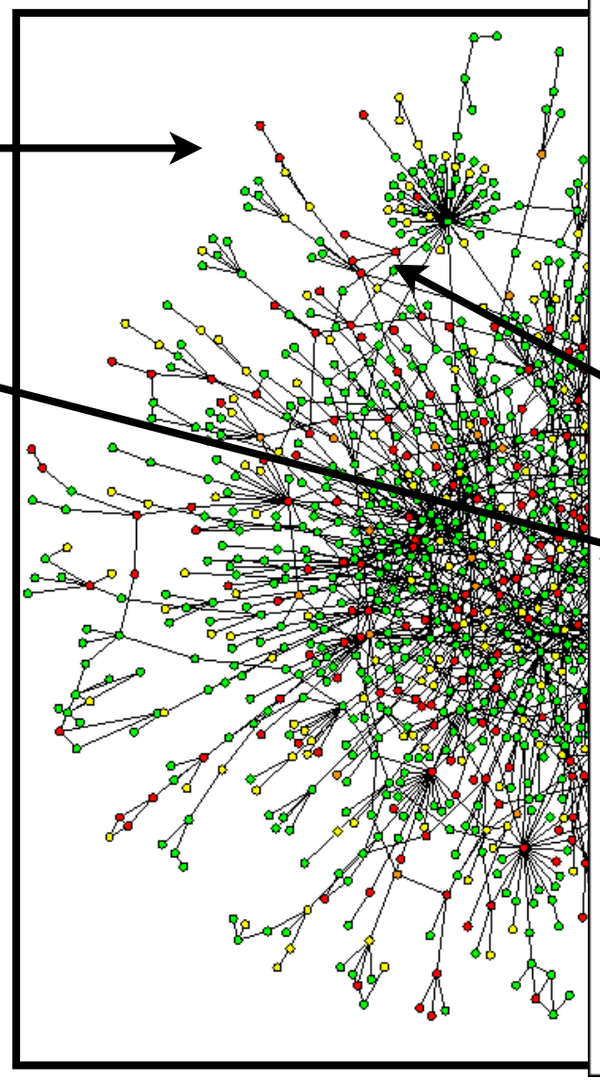
gene regulatory networks

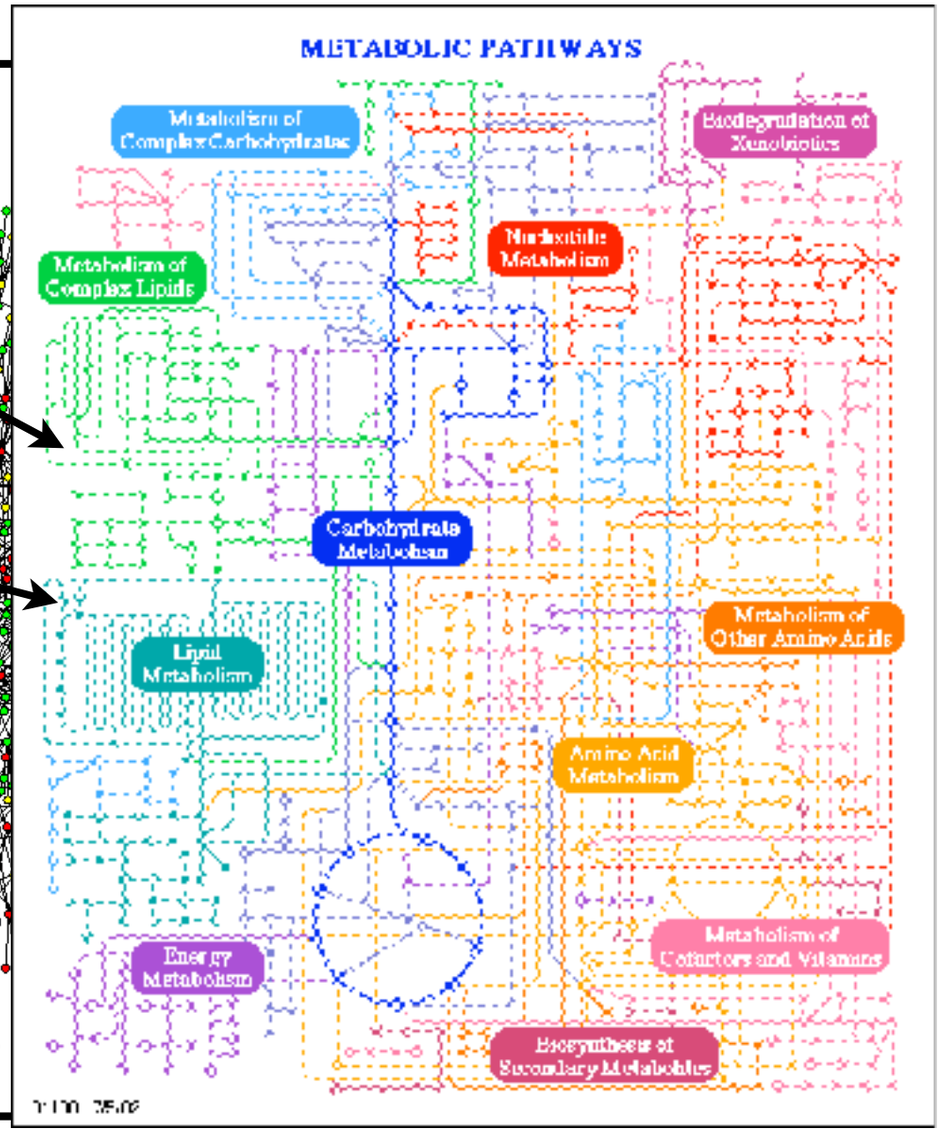
Everything is a network



gene regulatory networks



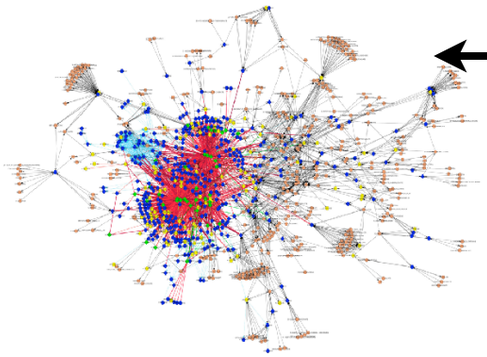




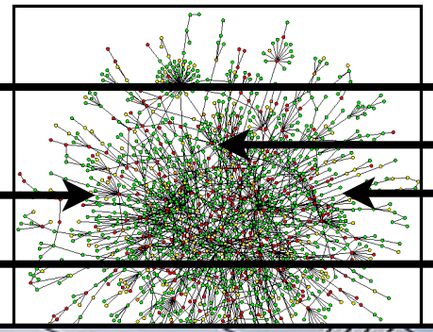
protein interaction networks

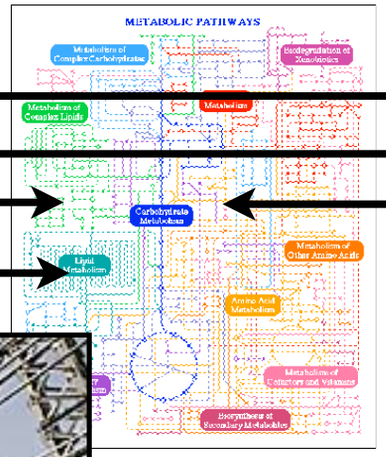
metabolic networks

Everything is a network

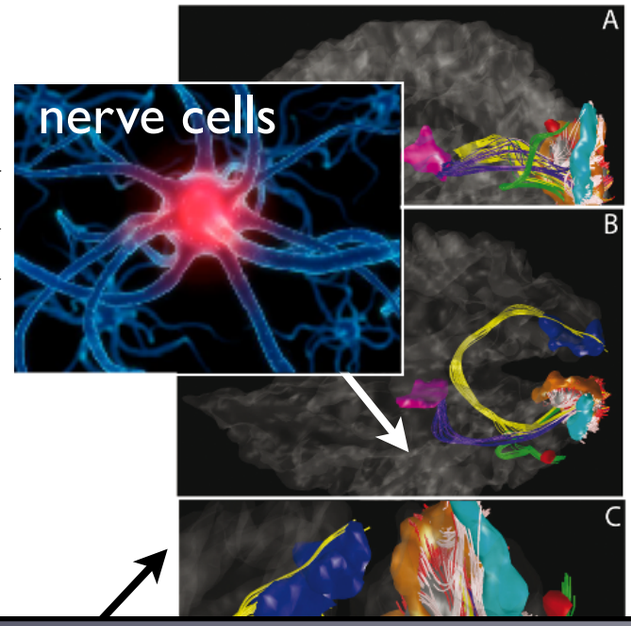


gene regulatory networks

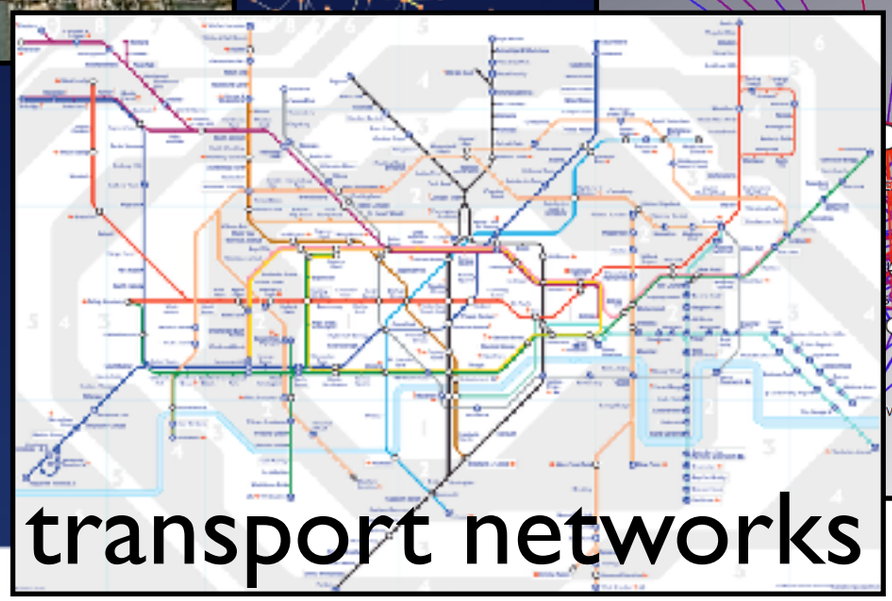


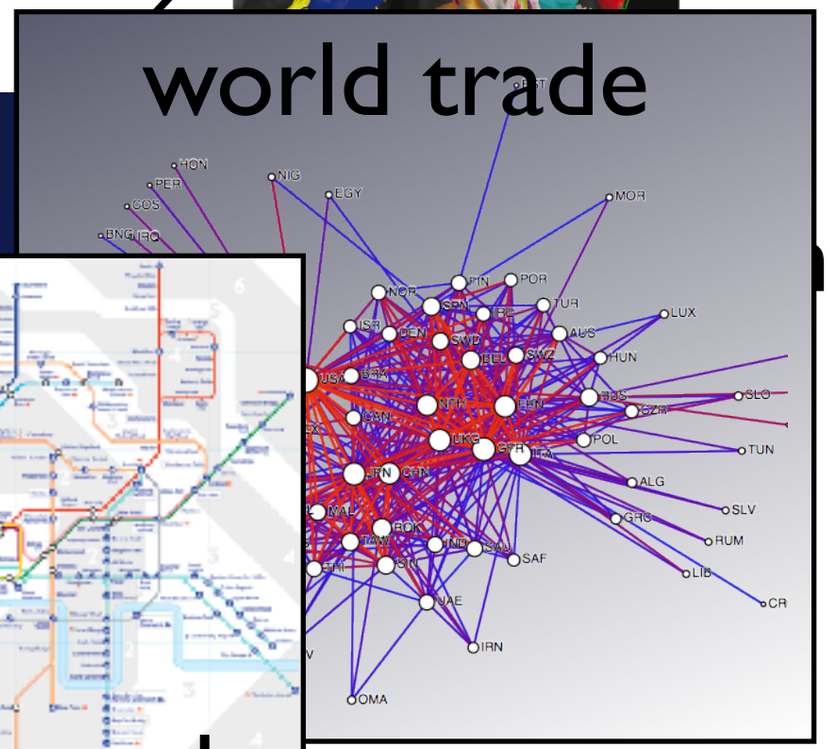


etabolic networks



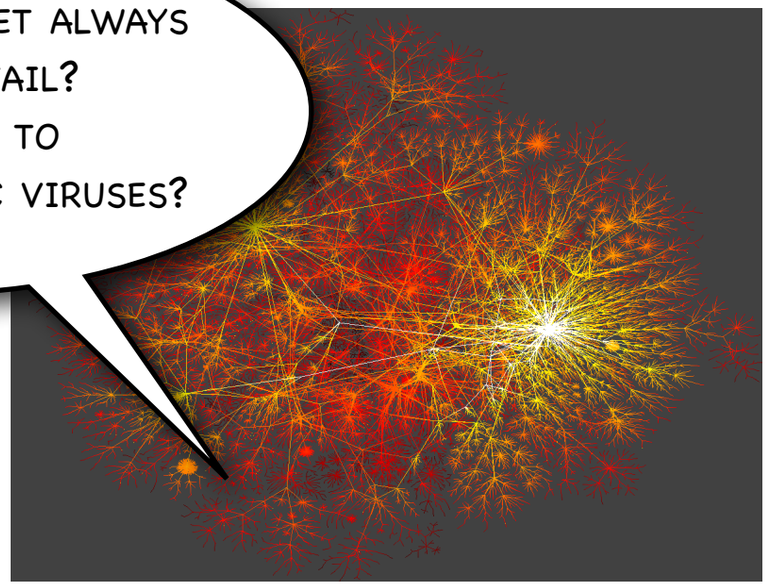
systems





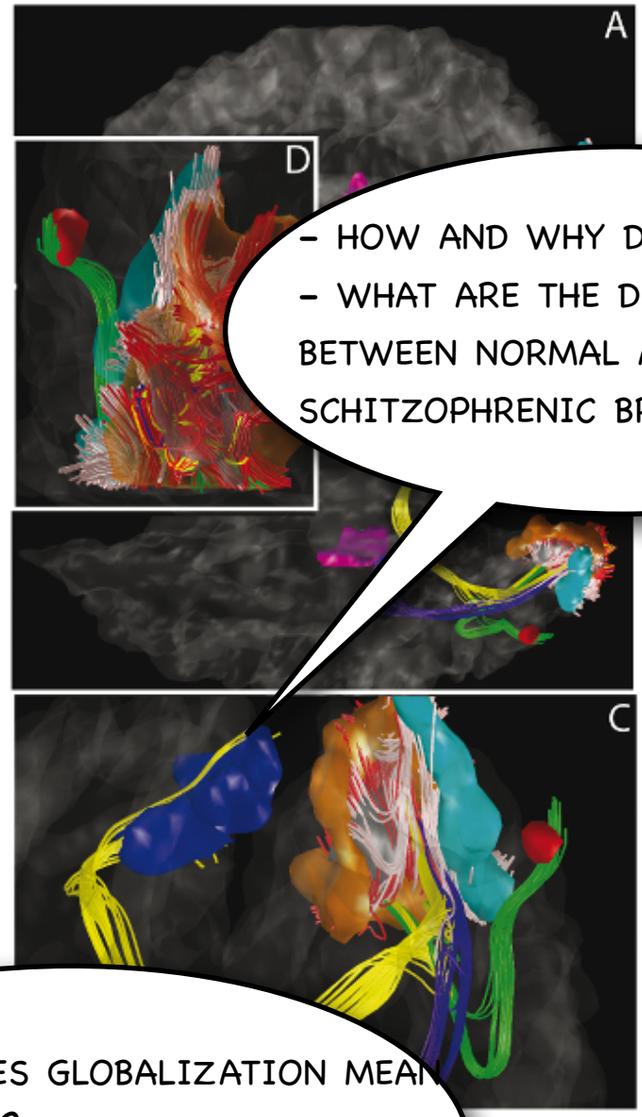
What can network science tell us?

Internet



- WHY IS THE INTERNET ALWAYS ON, WHY DOESN'T IT FAIL?
- WHY IS IT SO HARD TO ERADICATE ELECTRONIC VIRUSES?

human brain



- HOW AND WHY DO WE THINK?
- WHAT ARE THE DIFFERENCES BETWEEN NORMAL AND SCHITZOPHRENIC BRAINS?

world trade

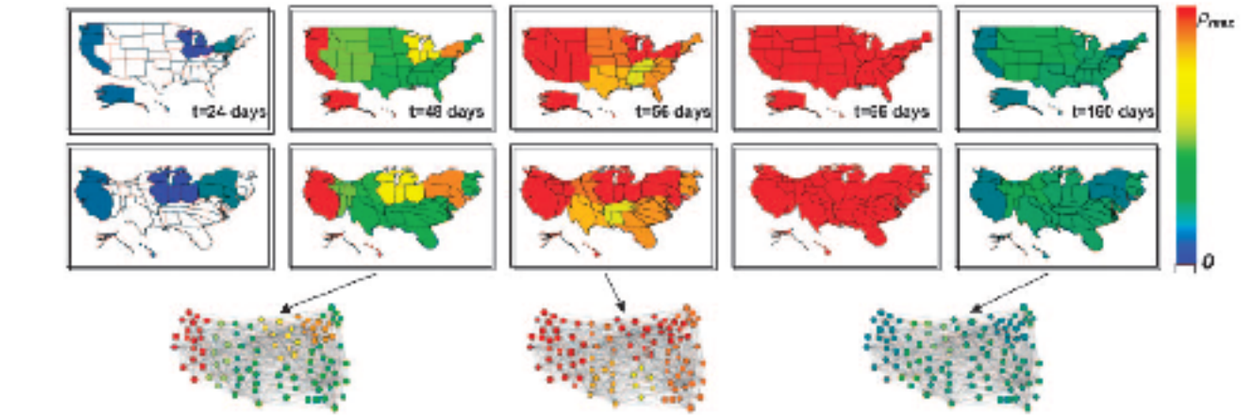
- WHAT DOES GLOBALIZATION MEAN IN PRACTICE?
- HOW CAN DEVELOPING COUNTRIES DEVELOP FURTHER?

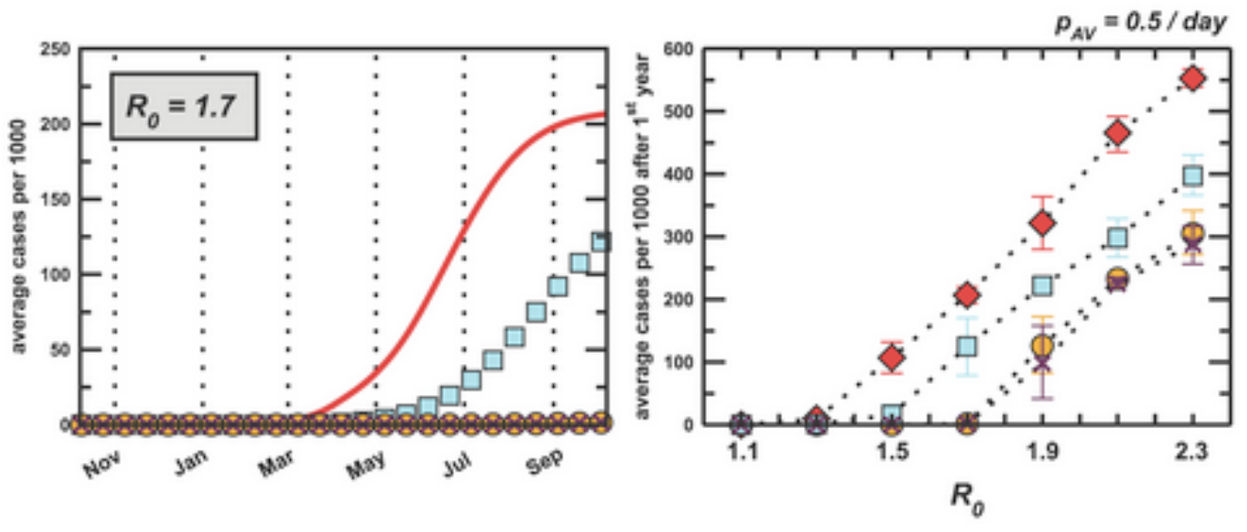
Hagmann P, Kurant M, Gigandet X, Thiran P, Wedeen VJ, et al. (2007) Mapping Human Whole-Brain Structural Networks with Diffusion MRI. PLoS ONE 2 (7): e597.

What can network science tell us?

-HOW DO DISEASES SPREAD?
-WHAT IS THE ROLE OF THE UNDERLYING CONTACT AND TRANSPORT NETWORKS?
-HOW TO PREVENT GLOBAL EPIDEMICS?

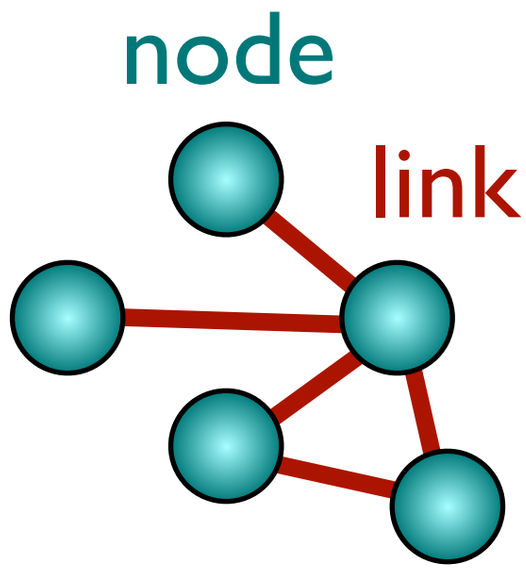
disease spreading





Complex systems as networks

- Links denote interactions between nodes
  - Interactions of different strength ⇒ weighted networks
  - Interactions of different direction ⇒ directed networks
  - Time-dependent interactions ⇒ temporal networks
  - Interactions of different type ⇒ multiplex networks
- Vertex = node (synonyms), edge = link



Vertex	Edge
person	friendship
neuron	synapse
WWW	hyperlink
company	ownership
gene	regulation

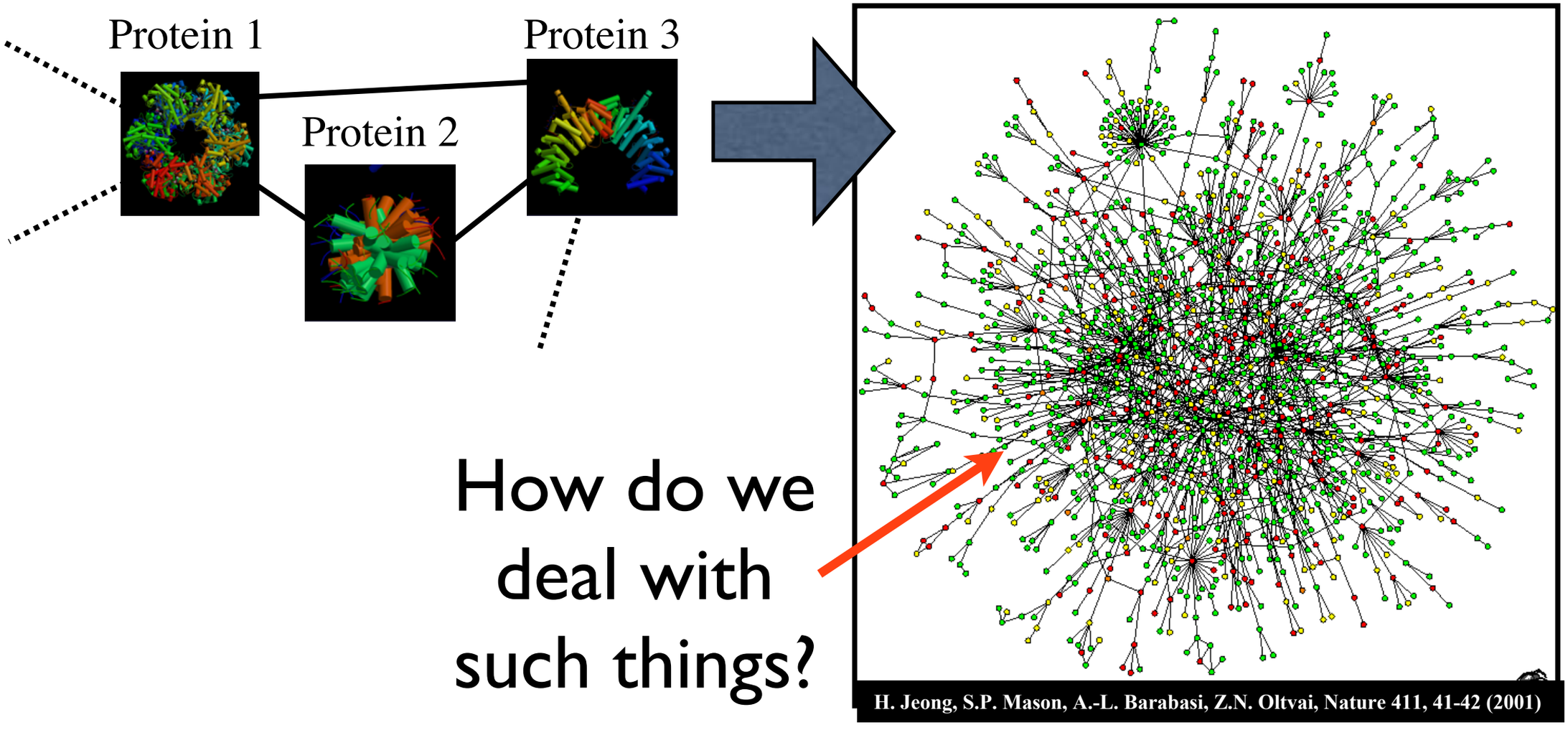
What is a link?
...that is not always straightforward.

Similarities			Social Relations				Interactions	Flows
Location	**Membership**	**Attribute**	**Kinship**	**Other role**	**Affective**	**Cognitive**	e.g.,	e.g.,
e.g.,	e.g.,	e.g.,	e.g.,	e.g.,	e.g.,	e.g.,	Sex with	Information
Same spatial and temporal space	Same clubs	Same gender	Mother of	Friend of	Likes	Knows	Talked to	Beliefs
	Same events	Same attitude	Sibling of	Boss of	Hates	Knows about	Advice to	Personnel
	etc.	etc.		Student of	etc.	Sees as happy	Helped	Resources
				Competitor of			Harmed	etc.
						etc.	etc.	

Borgatti et al., *Network analysis in the social sciences*. Science 323, 892-895 (2009).

Multiple types of links can be included using *multilayer networks* or *multiplex networks*. (More on these in the last lecture.)

The real question:



How do we deal with such things?

The network approach

1) Make empirical observations

2) Try to explain observations

2.1) Choose right level of coarse-graining
(interacting elements = nodes, interactions = links)

2.2) Strip the problem, disregard some detail
(assume the node-link structure contains the answers)

2.3) Cast the problem as maths
(analyze networks, simulate processes, write theories)

3) See if theories, calculations, or simulations can reproduce findings, or predict something

4) Start over from 1), refine

Part II:
Basic concepts

What is a network/graph?

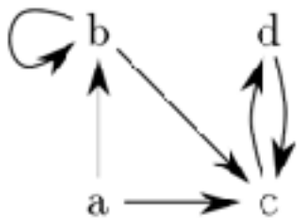
Graph $G = (V, E)$ consists of

- N vertices $V = \{v_1, \ldots, v_N\}$,
- m edges $E = \{e_1, \ldots, e_m\}$, where each edge is a pair of vertices, $e_i = (v_j, v_k)$.

If the vertex pairs (v_j, v_k) are ordered, the network is *directed*. Then $e_i = (v_j, v_k)$ means that there is an edge from v_j to v_k.

Otherwise the network is *undirected* and $e_i = (v_j, v_k) = (v_k, v_j)$ means that v_j and v_k are connected

Undirected graph with
$V = \{a, b, c, d\}$ and
$E = \{ab, ac, cb, cd\}$.



Directed graph with
$V = \{a, b, c, d\}$ and
$E = \{ab, ac, bc, bb, cd, dc\}$.

Note: link = edge, node = vertex, network = graph.
Choose whichever you like.

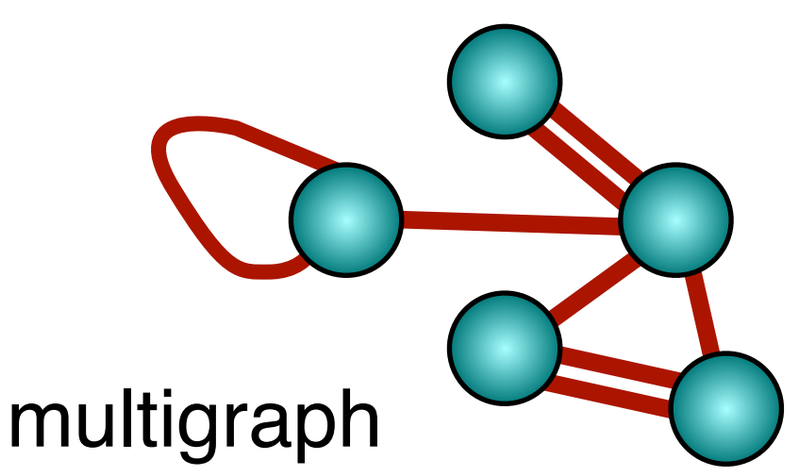
Simple graphs, multigraphs

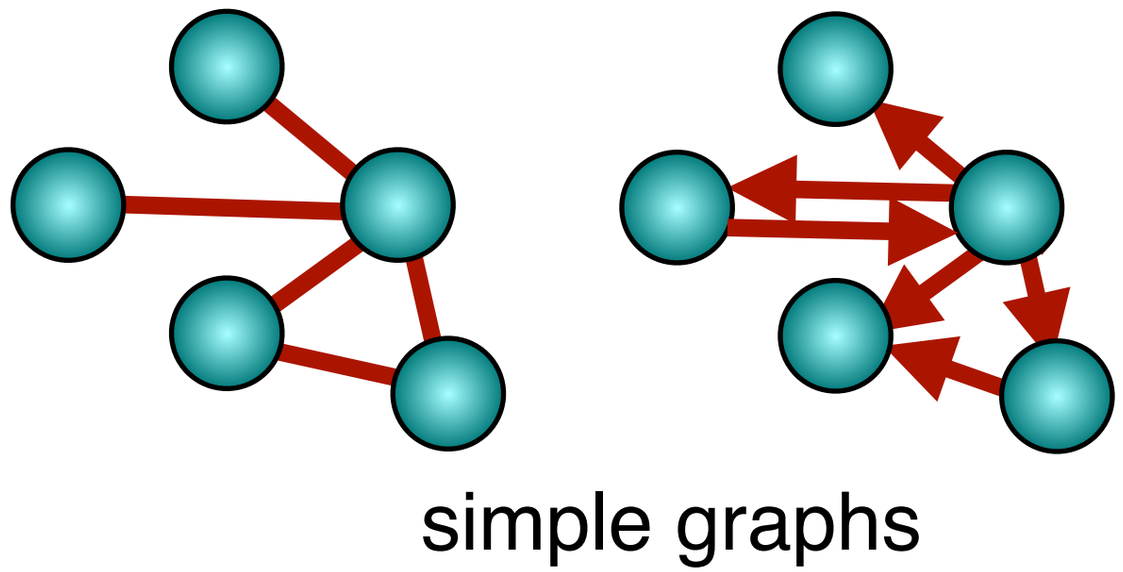
A *simple graph* has no self-loops, that is, links from a node to itself, or multiple edges between the same pair of nodes.

(Please note that in directed networks $(v_i, v_j) \neq (v_j, v_i)$ so bidirectional links do not count.)

Otherwise the graph is a *multigraph*.

In this course we only deal with simple graphs.

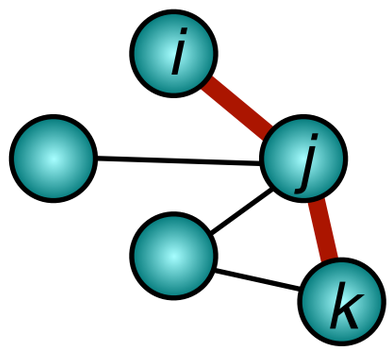




Walks and paths

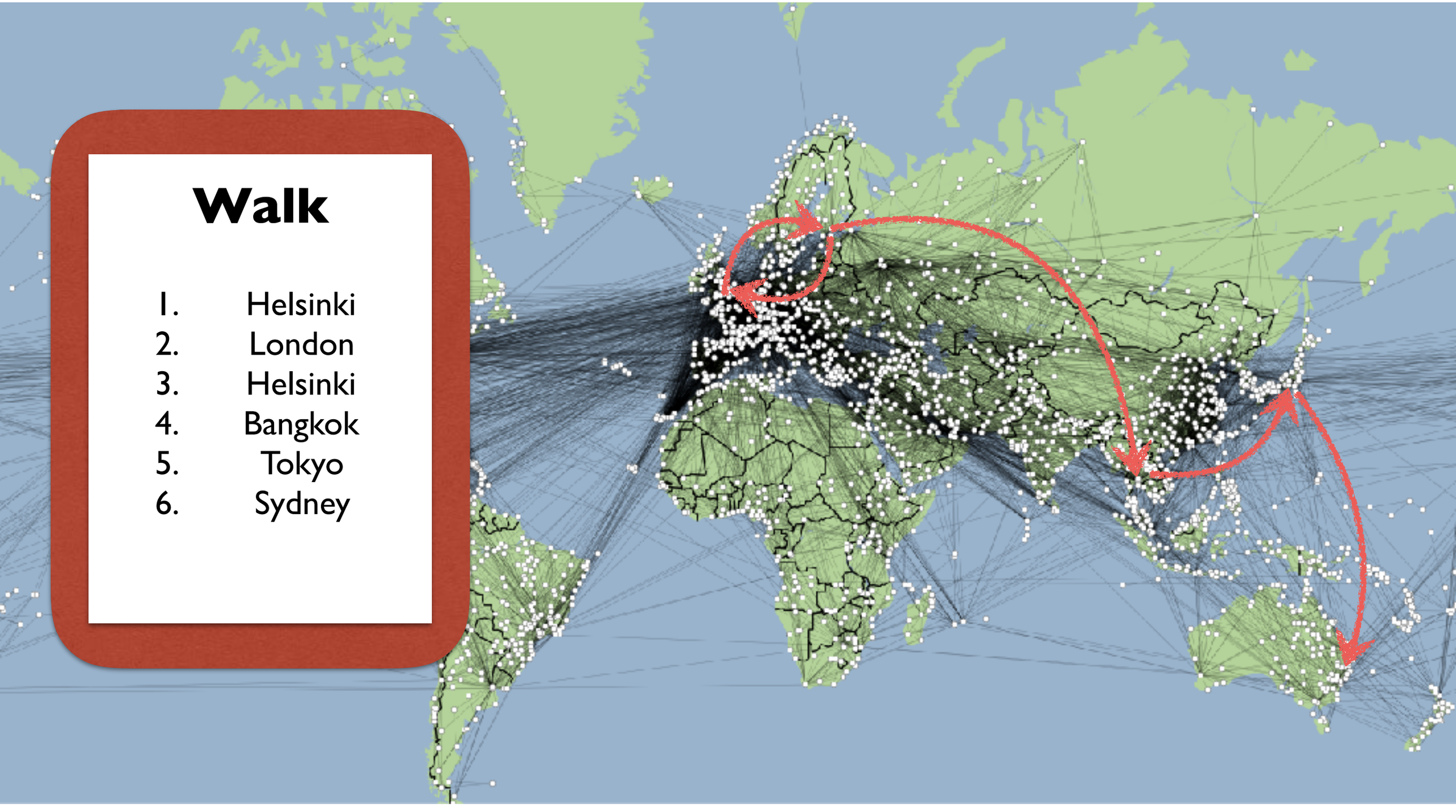
Naming conventions for walks and paths are varying (alternative names in parenthesis)

- *walk* (path) is a sequence of vertices where each consecutive pair is connected by an edge.
- *path* (self-avoiding walk/path, simple path) is a walk where vertices are never repeated. (Exception: the first and the last vertex can be the same.)
- *path length* is the number of edges traversed along a path.
- *shortest path* (geodesic path) is a path between a pair of nodes v_i and v_j with minimum path length.
- *distance* (geodesic distance), d_{ij}, is the shortest path length between nodes v_i and v_j.
- *diameter*, d, is the largest distance in the network: $d = \max_{i,j \in V} d_{ij}$.



Path {i,j,k} has length 2. This is the distance between i and k, and also happens to be the diameter of this network

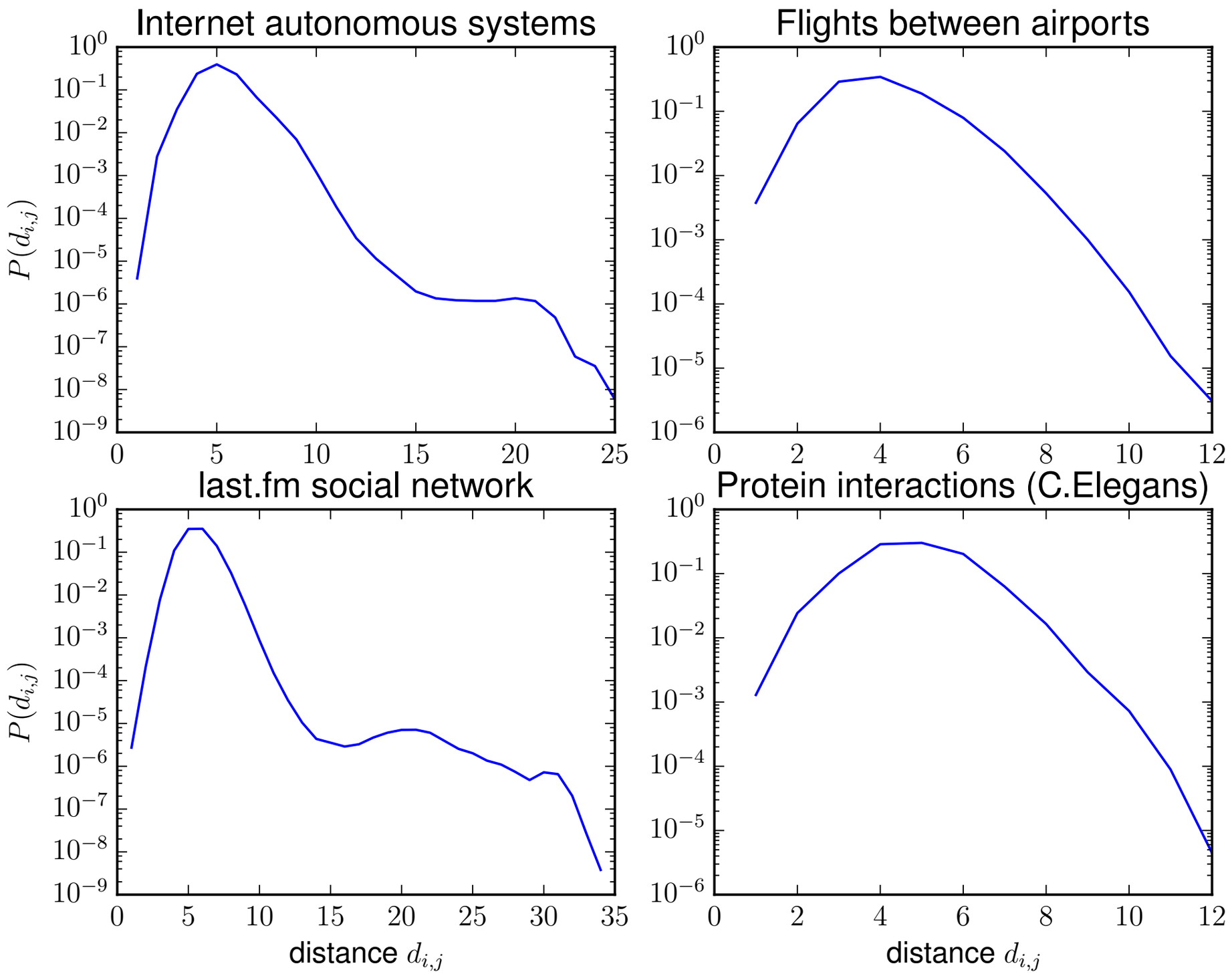
Examples in airline networks



Examples in airline networks

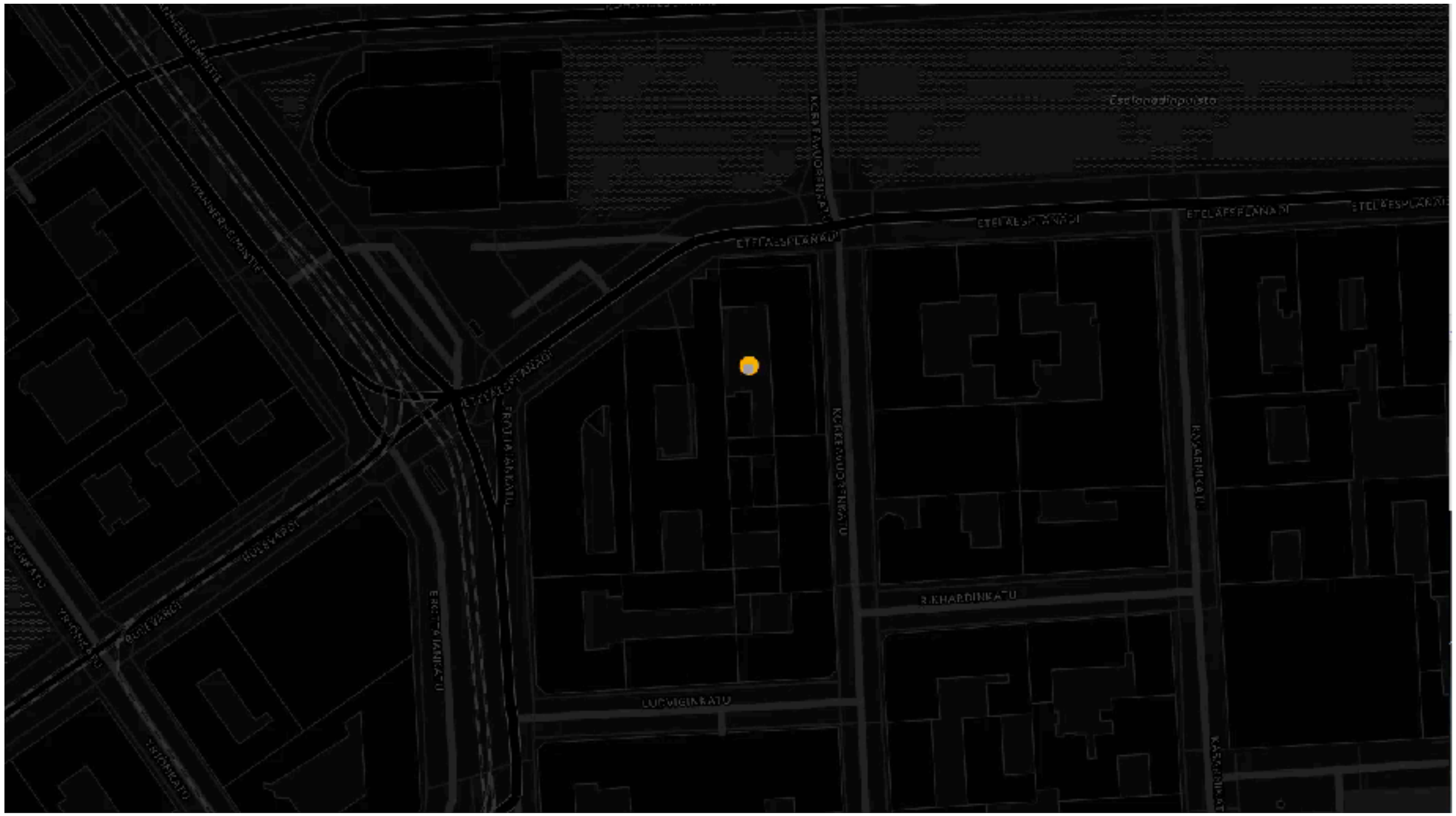
Examples in airline networks

Distances in data



- Large networks, but short distances
- Explanation in lecture 2

Example: shortest paths in the Helsinki public transport system

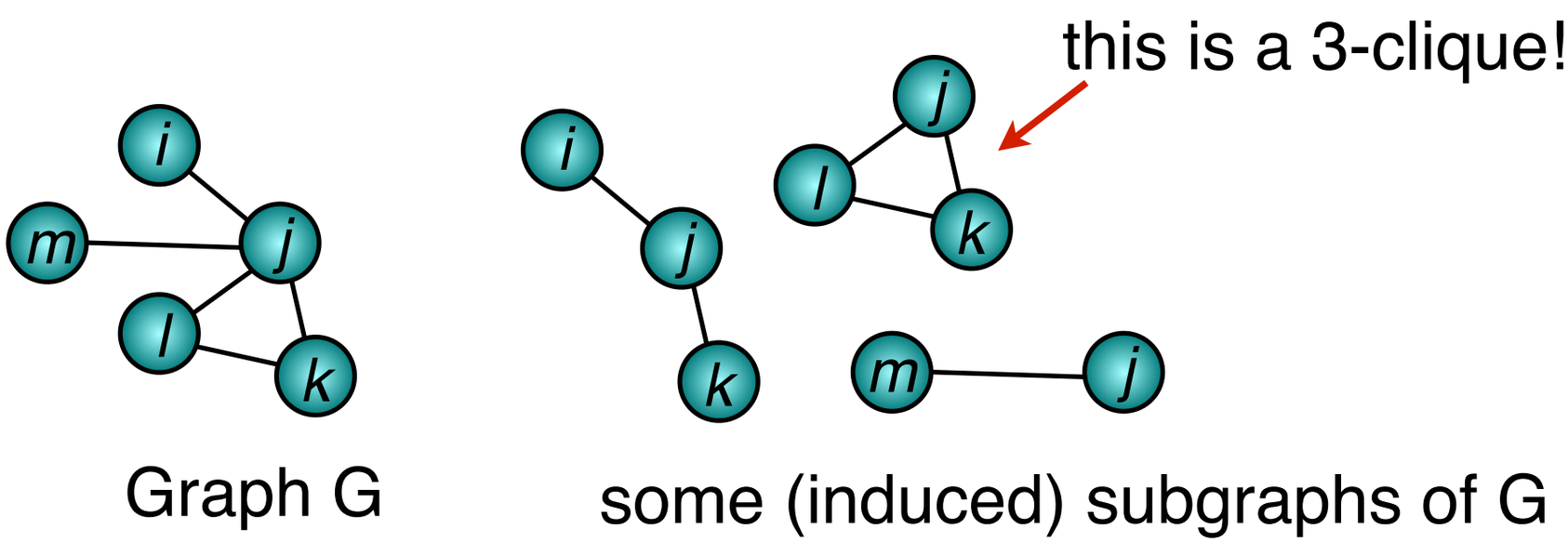


Subgraphs, cliques

A *subgraph* $G^* = (V^*, E^*)$ consists of some subset of nodes, $V^* \subseteq V$, together with some subset of edges between those nodes. *Induced subgraph* contains **all** edges between the nodes V^*.

$$\begin{aligned} \text{Subgraph: } E^* &\subseteq \{(v_i, v_j) \in E \,|\, v_i, v_j \in V^*\} \\ \text{Induced subgraph: } E^* &= \{(v_i, v_j) \in E \,|\, v_i, v_j \in V^*\} \end{aligned}$$

A clique is a subgraph where all nodes are linked to all other nodes.

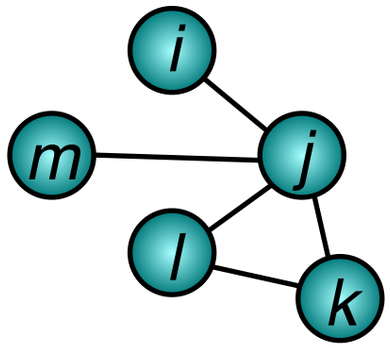


Connectedness, components

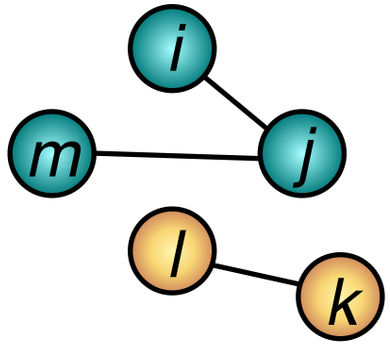
A graph is *connected* if some path can be found between all pairs of vertices.

If a graph is not connected, it consists of separate *components* (maximal connected subgraphs).

Note that because there is no path between nodes that belong to different components, their distance is undefined (or infinite).

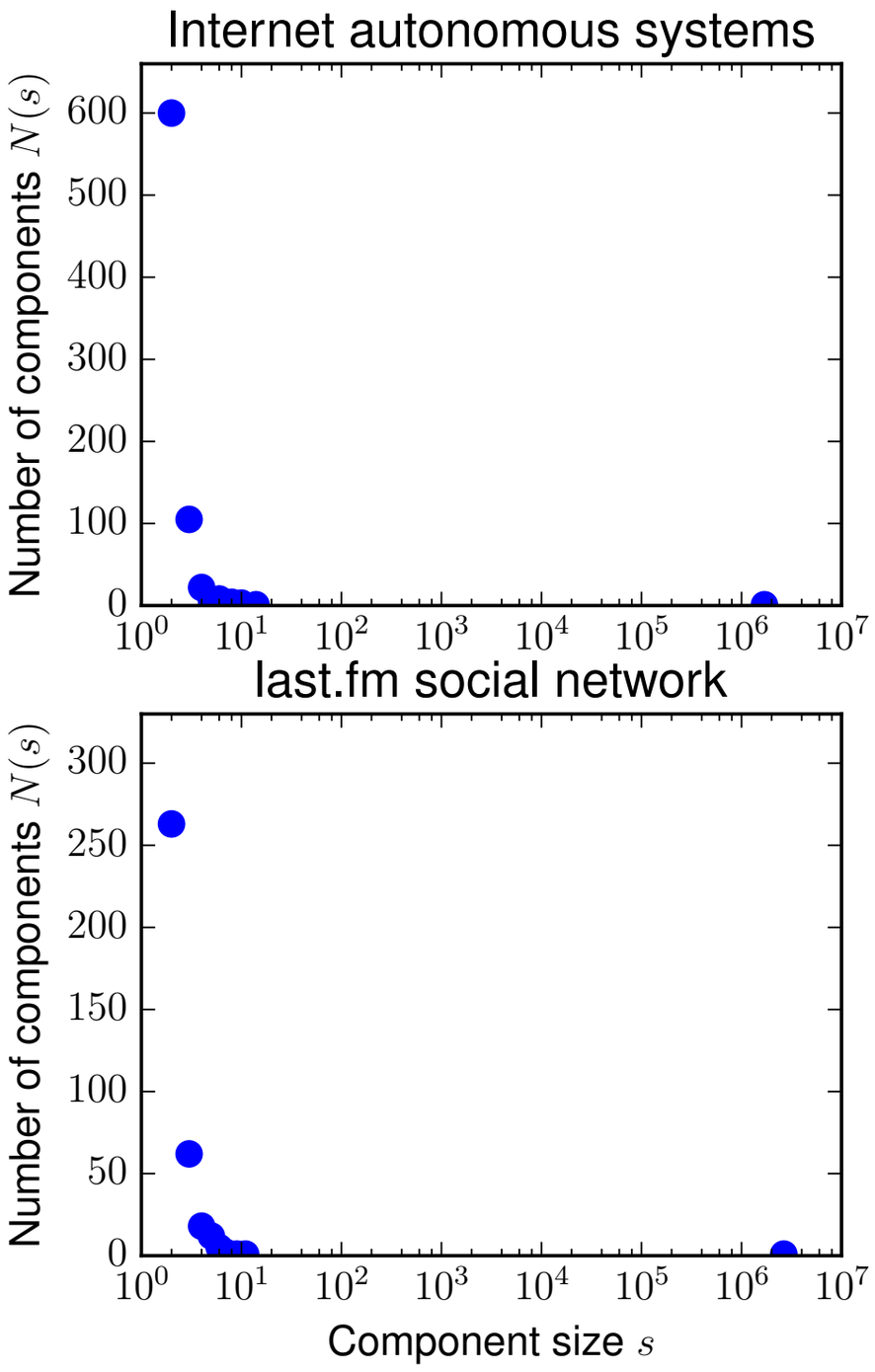


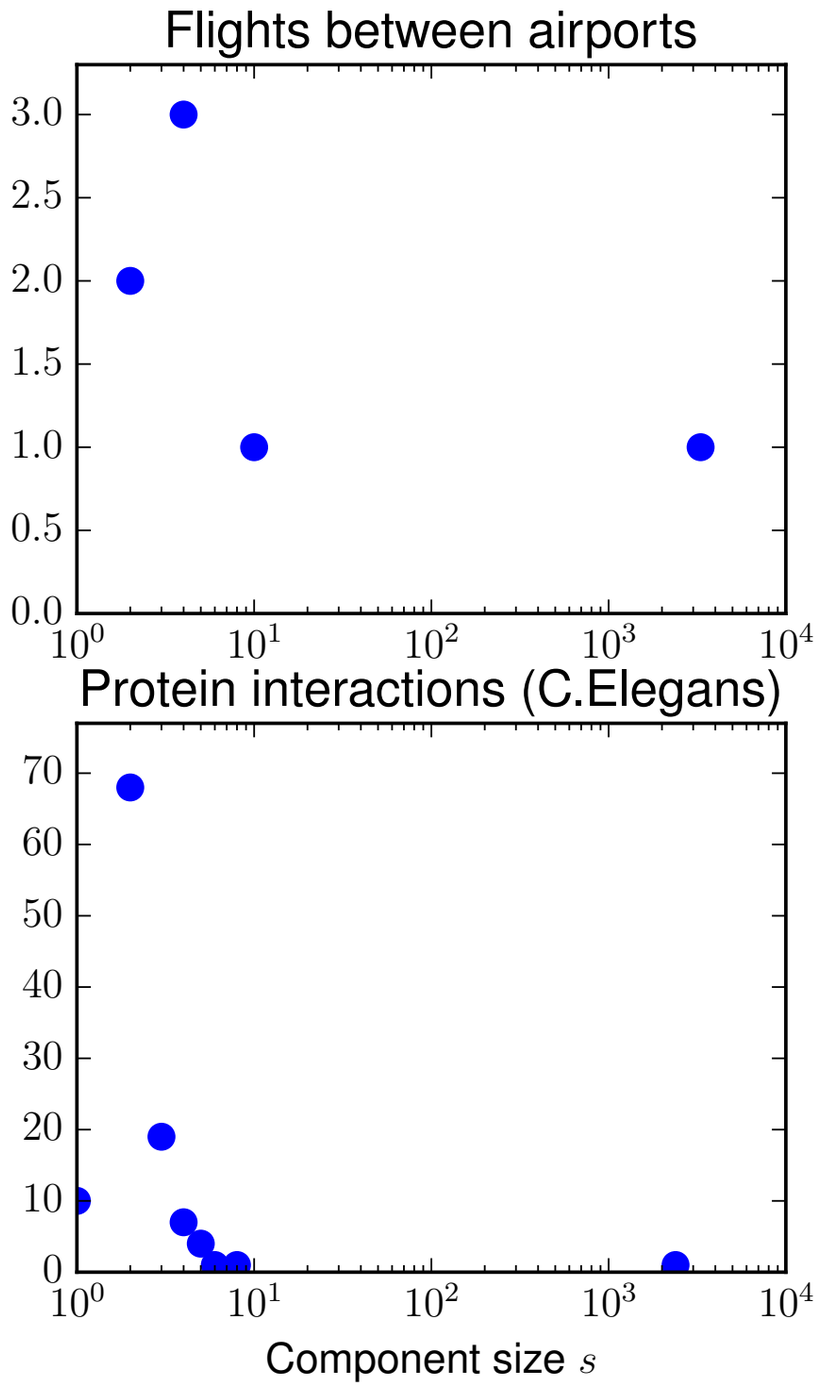
this graph is connected



this graph has two components, (i,j,m) and (l,k)

Components in data



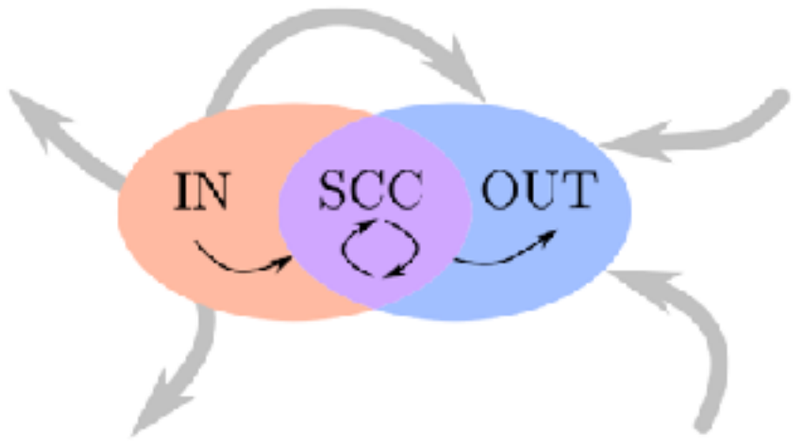


- Many small components
- One "giant" component
- Explanation in lecture 4

Directed networks: paths, components

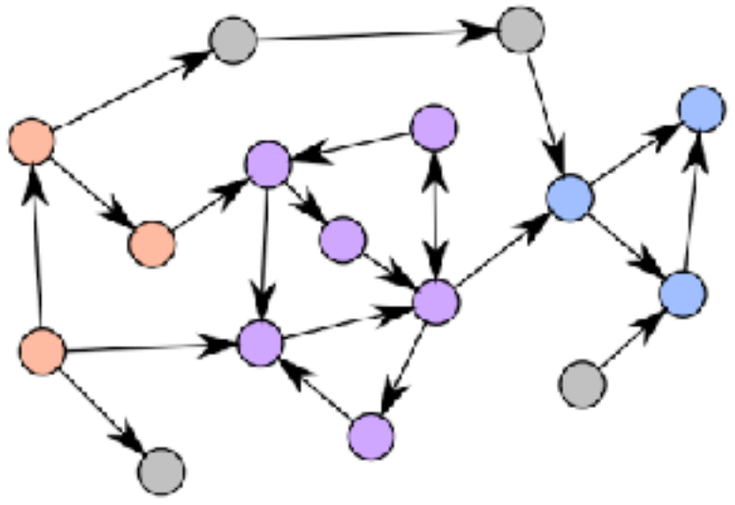
Directed paths

- Paths are usually required to traverse each edge according to its direction.
- We may have a path $v_i \to v_j$ but no $v_i \leftarrow v_j$, hence ...



Components

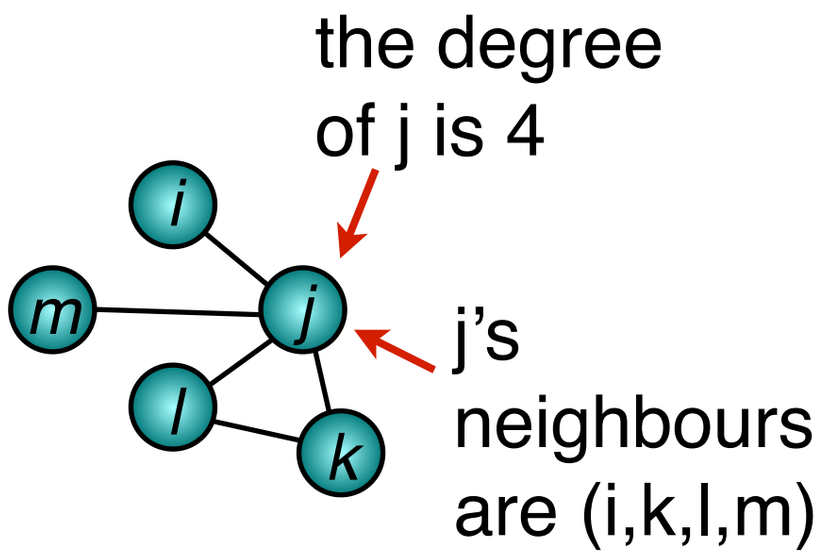
- Nodes in the **strongly connected component** (**SCC**) are mutually reachable.
- Nodes in the **IN-component** have paths to the SCC.
- Nodes in the **OUT-component** can be reached from the SCC.



Degree

If there is an edge $(v_i, v_j) \in E$,

- v_i and v_j are *adjacent*,
- v_i is a *neighbour* of v_j,
- the edge is *incident* to v_i and v_j.



The *degree* k_i of vertex v_i is the number of edges it is incident to. (This is number of neighbours in simple graphs. Loops are counted twice in multigraphs.)

For directed networks, one can consider separate *in-* and *out-degrees*. The in-degree $k_{i,in}$ is defined as the number of edges leading to v_i and the out-degree k_i, out as the number of edges leading out from v_i.

The average degree $\langle k \rangle$ of a network is $\langle k \rangle = \sum_i k_i / N = 2m/N$.

Degree distribution

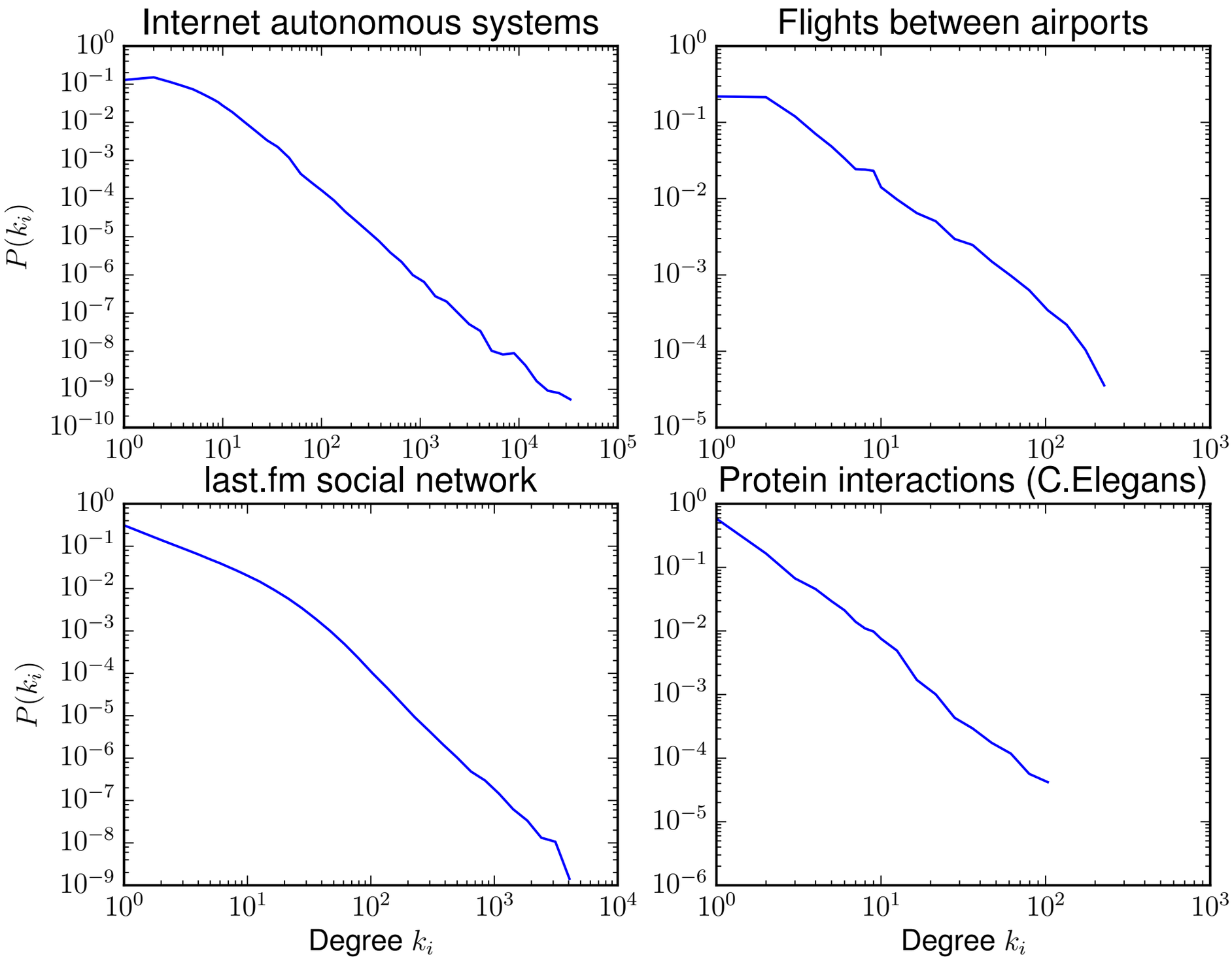
The degree distribution $P(k)$ is one of the central concepts in network analysis. It answers the questions "if a random node is picked, what is the probability that its degree is k?" That is,

$P(k) = N_k/N$,

where N_k is the number of nodes of degree k.

However, we often assume that the observed degree distribution is a sample from some "real" degree distribution that is smooth. In this case a better way to obtain an estimate for the real distribution is to *bin* the data and answer the question "what is the probability that a randomly picked node has a degree in some interval $[k_i, k_{i+1}]$?"

Degree examples



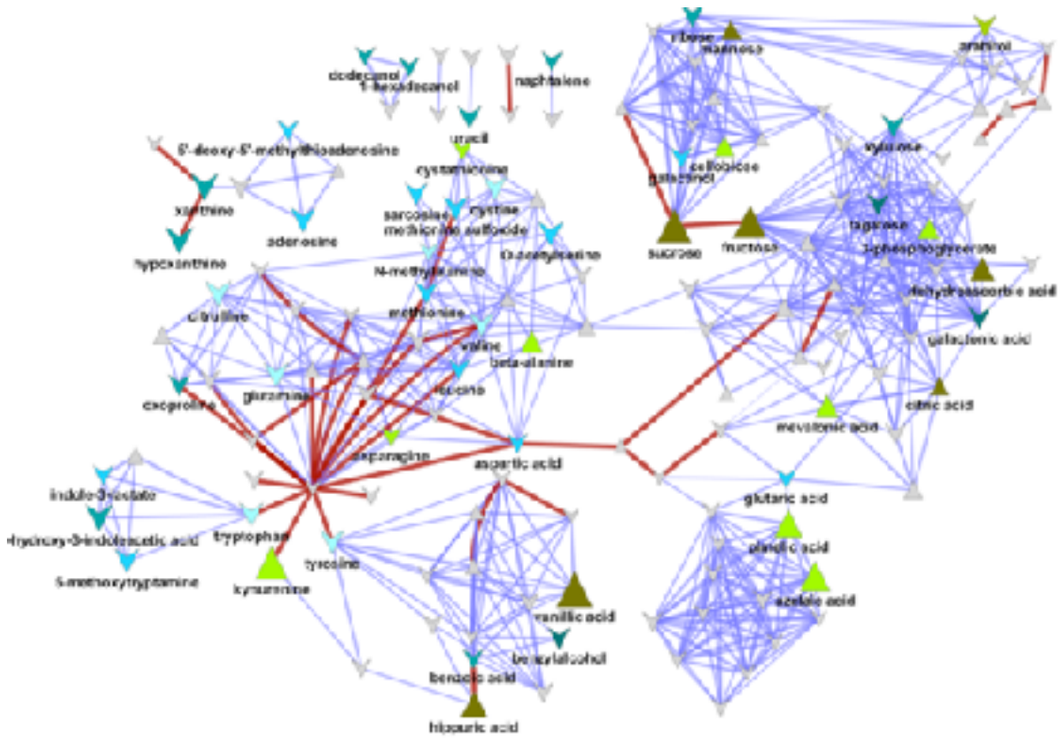
- Most nodes have very small degree, few "hubs"
- Explanation in lecture 3

Edge density

The edge density of a network is the fraction of edges out of possible edges:

$$\rho = \frac{m}{\binom{N}{2}} = \frac{2m}{N(N-1)}.$$

In real-world networks, the edge density is usually low, *i.e.* the networks are *sparse*.



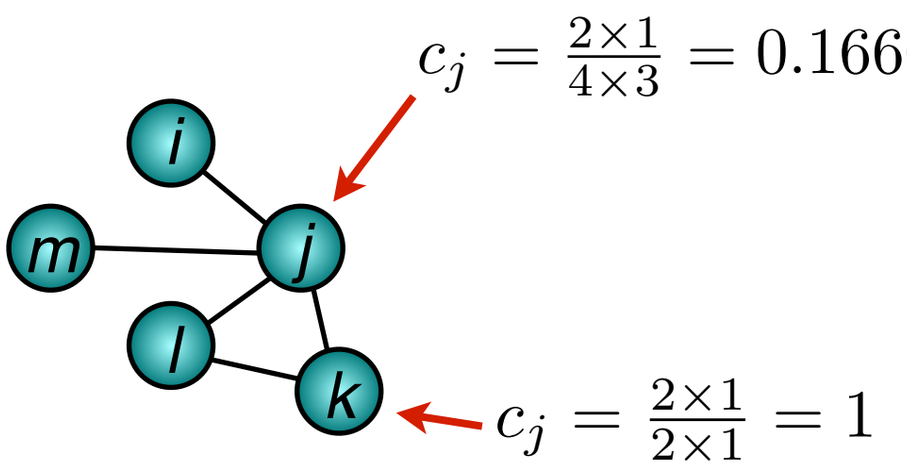
even though this metabolic network has dense subgraphs, in general it has a low link density, i.e. is sparse.

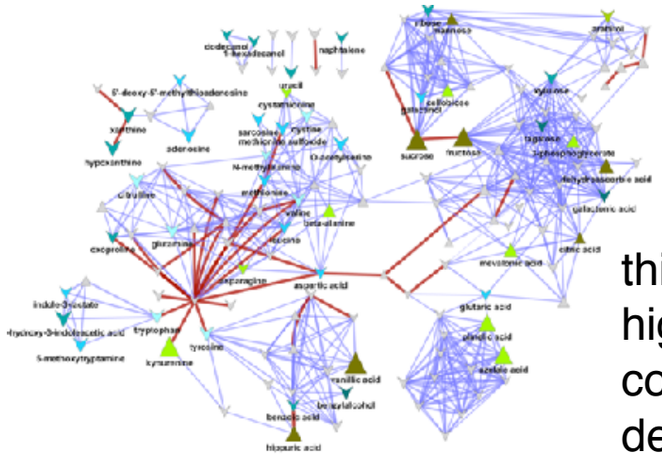
Clustering coefficient

Clustering coefficient defined for node i as the fraction of edges between its neighbours out of possible edges between its neighbours:

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

Here E is the number of edges between i's neighbours. Note that the clustering coefficient can be interpreted as density of i's neighbourhood.





this metabolic network has a high average clustering coefficient because of the dense subgraphs.

Clustering coefficient and transitivity

Alternative, and equivalent, definition:

$$c_i = \frac{E_i}{\binom{k_i}{2}} = \frac{\text{number of triangles centered at i}}{\text{number of 2-stars centered at i}}$$

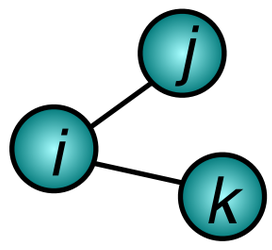
Globally one can measure *transitivity*:

$$C = \frac{\sum_i E_i}{\sum_i \binom{k_i}{2}} = \frac{\text{number of triangles in the network}}{\text{number of 2-stars in the network}}$$

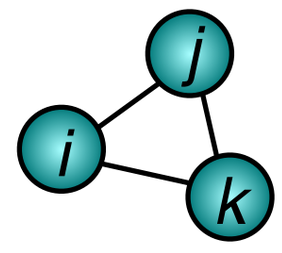
Note: transitivity is not the same as average clustering coefficient

$$c = \frac{1}{N}\sum_i c_i = \frac{1}{n}\sum_i \frac{E_i}{\binom{k_i}{2}} \neq C = \frac{\sum_i E_i}{\sum_i \binom{k_i}{2}}$$

2-star centered at i



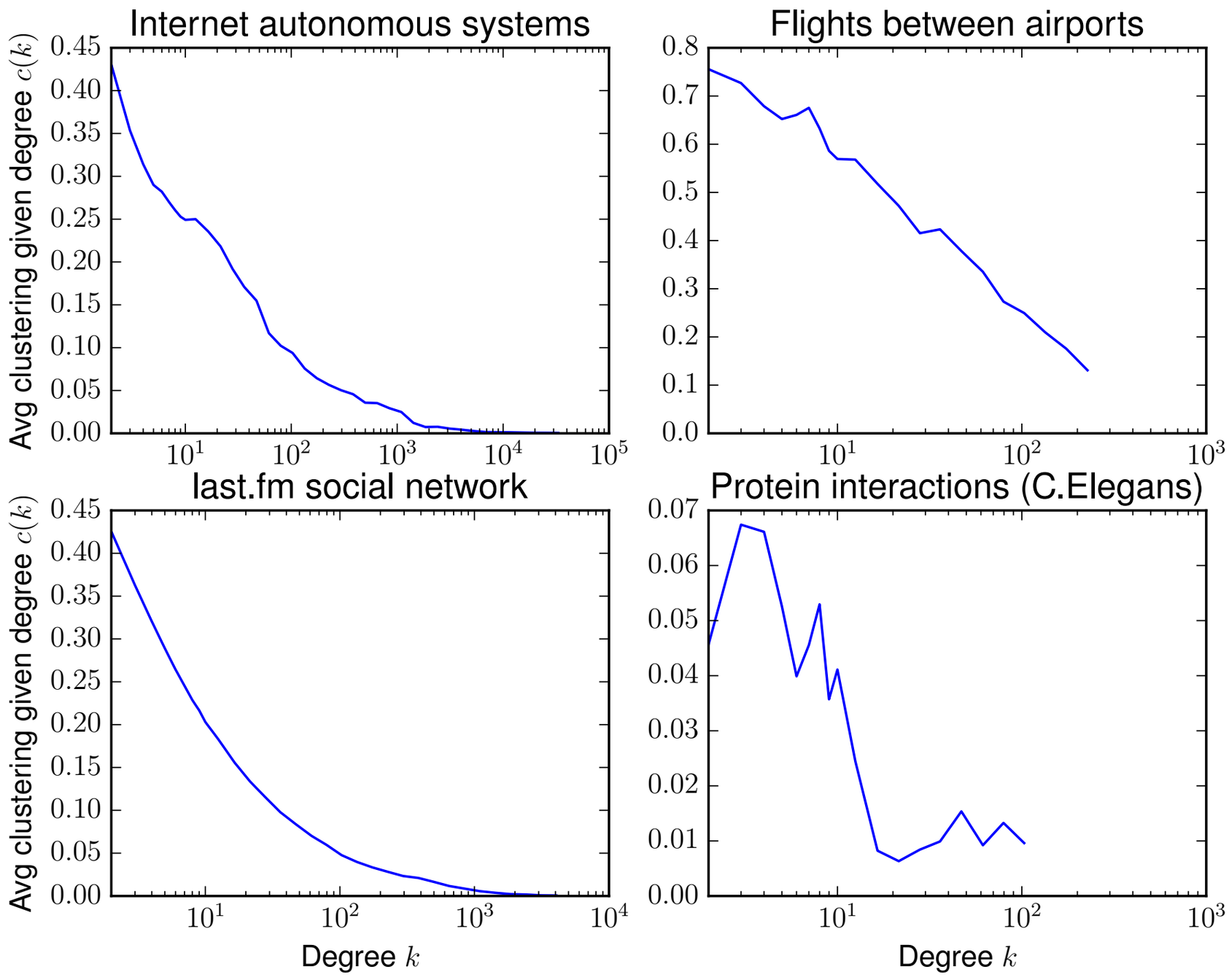
Triangle centered at i



Examples of values of density and clustering coefficient

Network	ρ	c	C
last.fm social net	3.2 10^{-6}	0.19	0.033
Airports & flights	0.0034	0.49	0.25
Protein interaction	0.0012	0.02	0.013
Internet AS	7.7 10^{-6}	0.25	0.0054

Examples of values of density and clustering coefficient



Special graphs

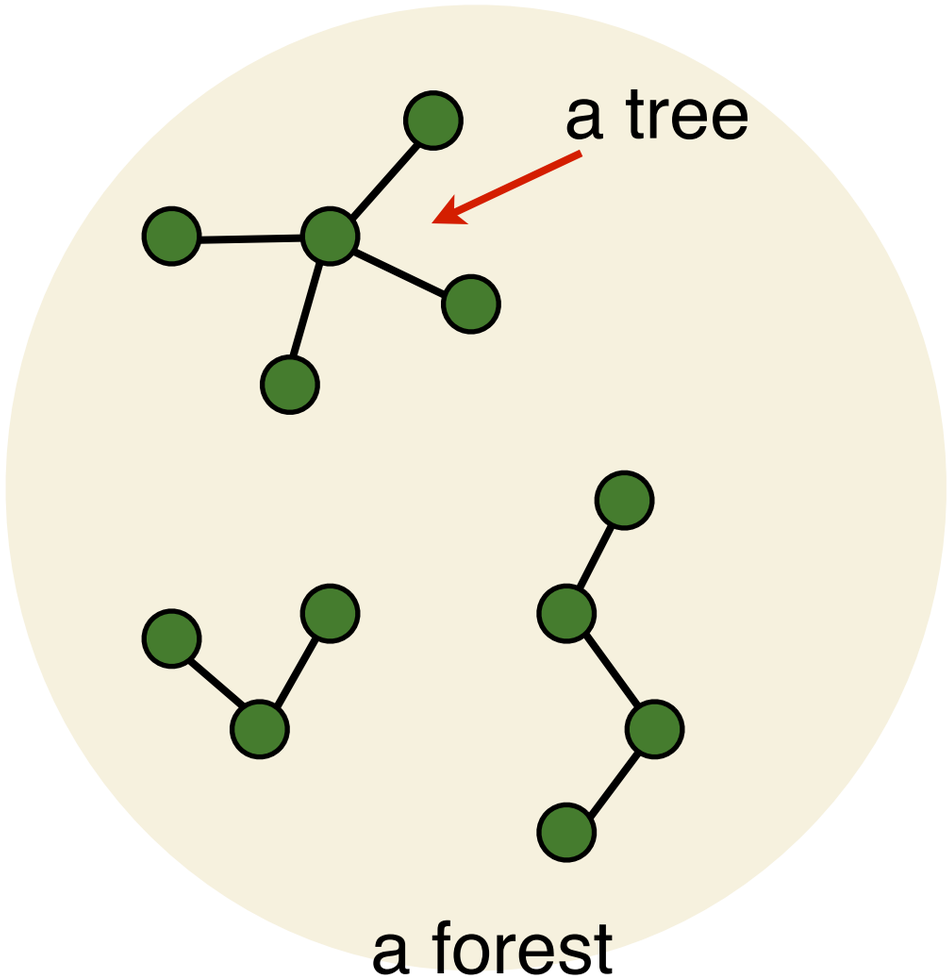
- A ***tree*** is a connected graph with no loops.

- So C_i=0 for all i, <C>=0.

- A tree with N vertices always has m=N-1edges.

- Also: a connected graph with m=N-1 edges is always a tree.

- A set of trees is called a ***forest.***

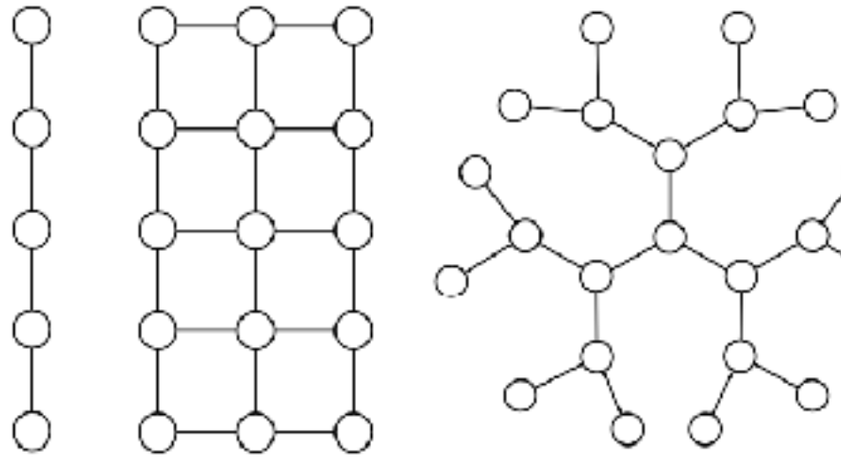


Special graphs

- In a ***k-regular graph*** all nodes have degree k.

- k-regular graphs can e.g. take the shape of regular lattices, or be otherwise random.

- ***Cayley tree***:
  a k-regular (infinite) tree.



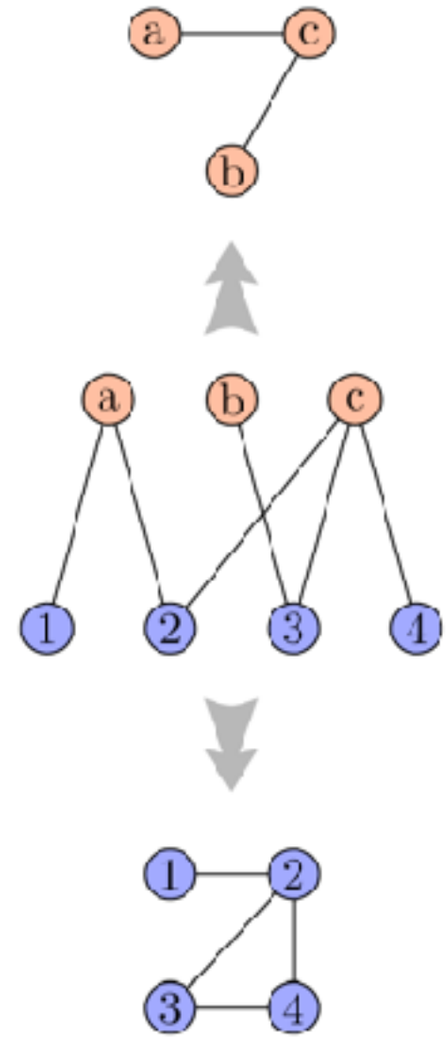
Subgraphs of 1-d lattice,
2-d lattice, and a Cayley tree

Bipartite graphs

- If the vertices of a graph can be divided into two subsets V_1 and V_2 such that edges exist only between subsets, the graph is ***bipartite***

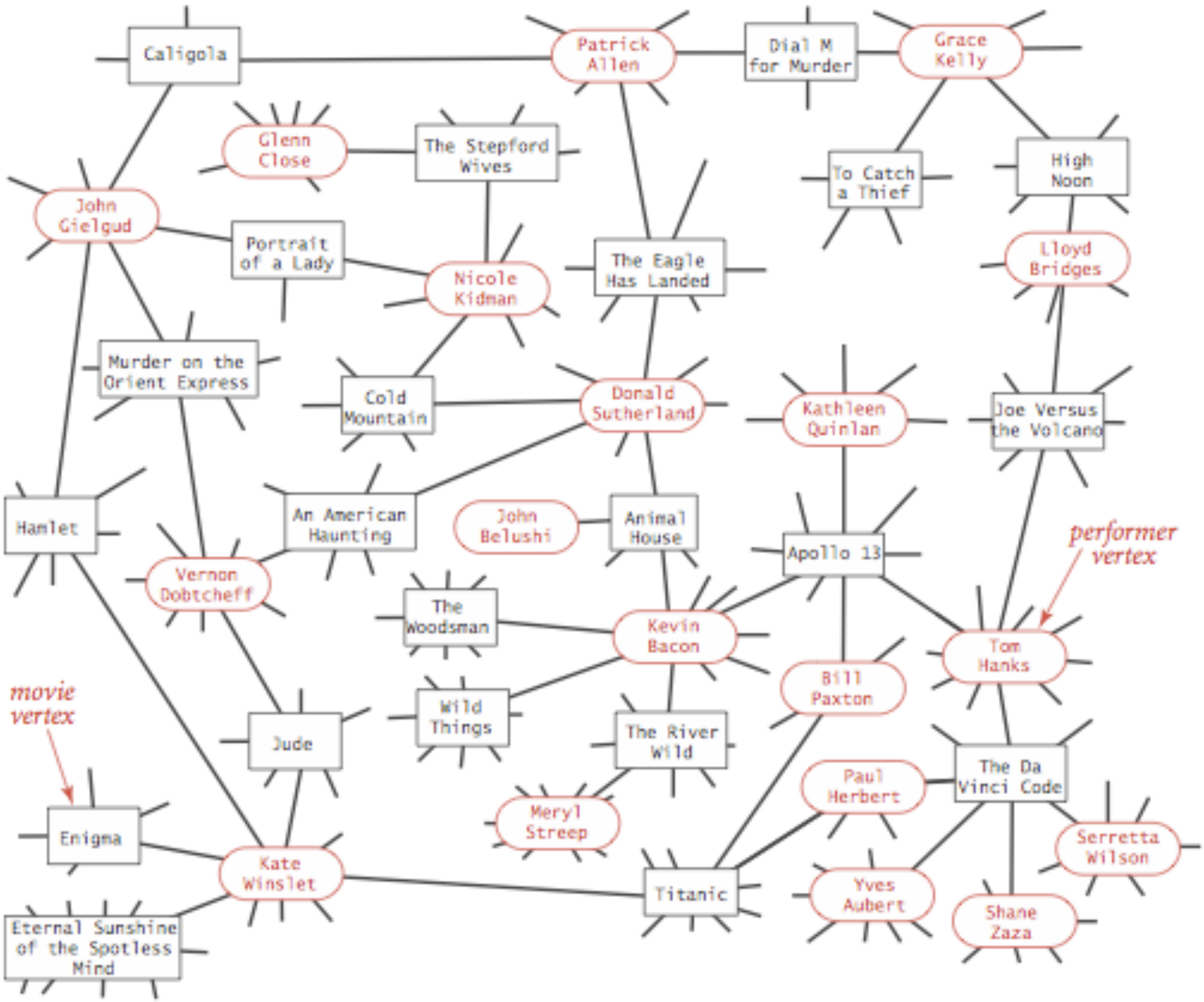
- A bipartite graph can be *projected* (or *collapsed*) onto V_1 or V_2

- Bipartite graphs arise naturally in many contexts (collaboration networks, metabolic reactions, etc)



A bipartite graph with $V_1 = \{a, b, c\}$, $V_2 = \{1, 2, 3, 4\}$ and the resulting projections to V_1 and V_2.

Example: actors & movies

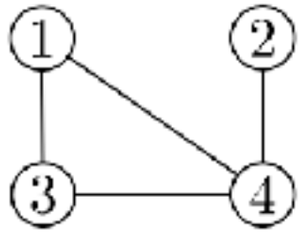


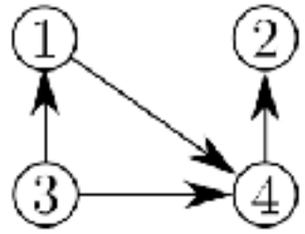
Network representation

Let us denote nodes by integers from 1 to N.

Graphical representation is informative for small graphs.

The network can be represented as an **edge list**, which is just a simple way to present E.

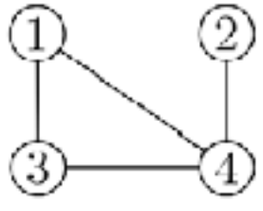


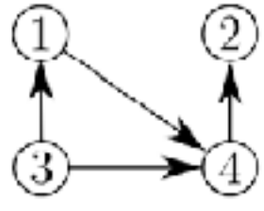


edge	i	j
1	1	3
2	1	4
3	3	4
4	2	4

edge	i	j
1	3	1
2	1	4
3	3	4
4	4	2

Network representation





The **adjacency list** gives the neighbors of each node.

i: neighbors
1: 3, 4
2: 4
3: 1, 4
4: 1, 2, 3

i: neighbors
1: 4
2:
3: 1, 4
4: 2

The $n \times n$ **adjacency matrix** A has elements a_{ij} defined by

$$a_{ij} = \begin{cases} 1 & if (j,i) \in E, \\ 0 & if (j,i) \notin E \end{cases}$$

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

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

$a_{ii} = 0$ for all simple graphs

Network data structures

Adjacency matrix

- Usable only for small networks.
- For example,
  - 32-bit integers for nodes
  - Network with $n = 10^5$, $m = 10^6$
  - Saving as edge list:

$$4 \cdot 2 \cdot 10^6 \approx 8 \text{ MB}$$

  - Saving as adjacency matrix?

$$4 \cdot 10^5 \cdot 10^5 \approx 40 \text{ GB}$$

Better ways

- Most real networks are sparse ($\rho << 1$)
- We only need to save the non-zero values (e.g. with the adjacency list)
- Edge list takes little space, but is inefficient.
- More efficient methods can be build using the adjacency list.

Networkx does everything automatically, so you do not have to worry about all this.

Just do not try to manually generate an adjacency matrix for any larger network...

More advanced network data structures

- Adjacency lists as sorted arrays

  + Minimal memory usage
  - Adding links O(k), querying connections O(log[k])
  - Used in: Snap, Graph-tool

- Adjacency lists as hash maps

  + All operations constant time on average
  - Slightly more space usage compared to arrays
  - Used in: Networkx, Lcelib

Adjacency matrix = not just data container, also powerful mathematical tool

undirected network

$$A = A^T$$

$$k_i = \sum_{j=1}^{N} a_{ij} = \sum_{i=1}^{N} a_{ij}$$

$$2m = \sum_{i=1}^{N} k_i = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}$$

directed network

$$k_{i,\text{in}} = \sum_{i=1}^{N} a_{ij}$$

$$k_{i,\text{out}} = \sum_{j=1}^{N} a_{ij}$$

$$m = \sum_{i=1}^{N} k_{i,\text{in}} = \sum_{i=1}^{N} k_{i,\text{out}}$$

Extra reading material

Books:

- **M.E.J. Newman: *Networks: an Introduction* (Oxford UP, 2018)** recommended!
- **A-L Barabási: *Network Science*** (Cambridge UP, 2016) (online at http://barabasi.com/networksciencebook/)
- D. Easley & J. Kleinberg, *Networks, Crowds, and Markets: Reasoning about a Highly Connected World* (Cambridge UP, 2010)
(online at http://www.cs.cornell.edu/home/kleinber/networks-book/)

Review papers:

- **M.E.J. Newman: *Structure and function of complex networks*** (SIAM Review 45, 167-256 (2003)) (online at http://arxiv.org/abs/cond-mat/0303516/)

recommended!

Extra reading material

Python and Networkx, online documentation and tutorials:

- http://docs.python.org/
- https://networkx.readthedocs.io/en/stable/

Other network software (not used in the course, but good to know):

- http://snap.stanford.edu/
- http://graph-tool.skewed.de/
- http://github.com/CxAalto/
- http://www.boost.org/doc/libs/1_61_0/libs/graph/
- http://igraph.org