An Introduction to Networks

Social Network Analysis with the “igraph" package in R

Amanda Perofsky

Introduction to Biological Statistics Workshop

November 6, 2015

Overview

Network basics/Mathematics of networks

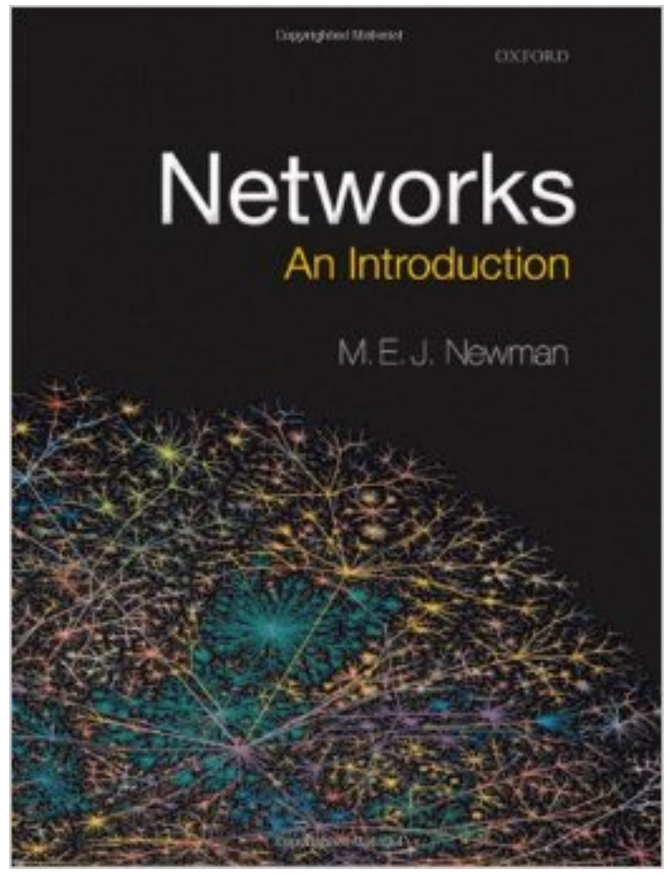
R package: “igraph”

Vertex centrality measures

Network-level measures of structure

Network models

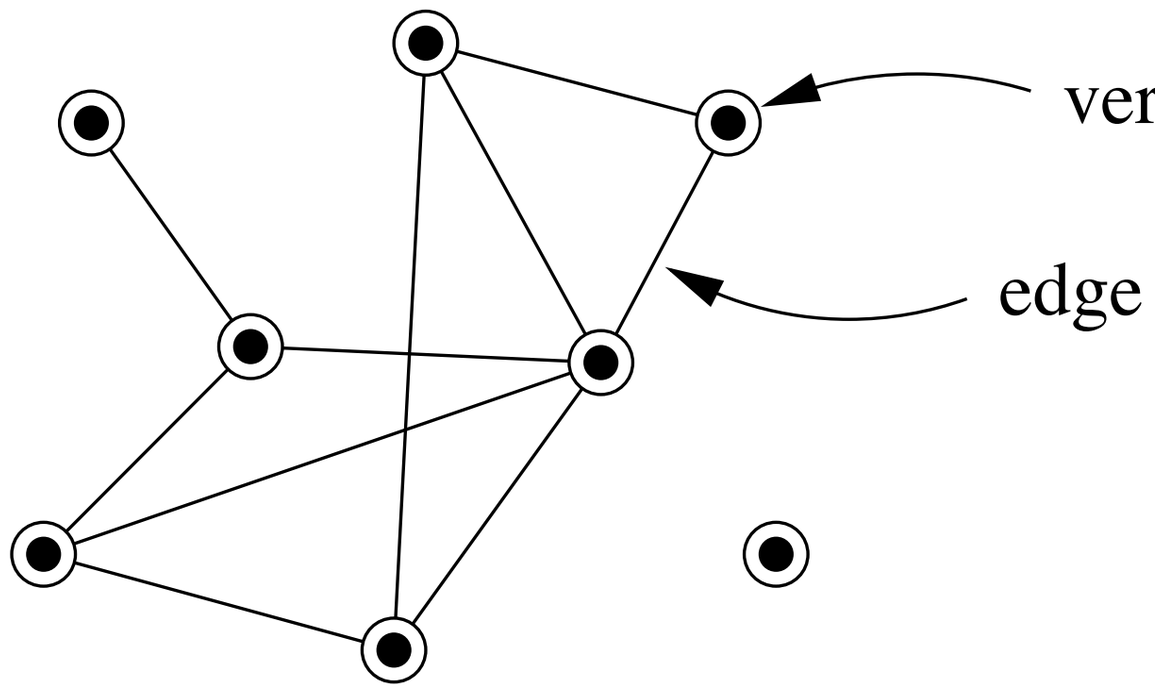
Acknowledgements



Dr. Lauren Ancel Meyers
Professor of Integrative Biology

What is a network?

A **network** is a collection of points, which we refer to as **vertices** or **nodes**, with connections between them, called **edges**.



In mathematics, these are called **graphs**.

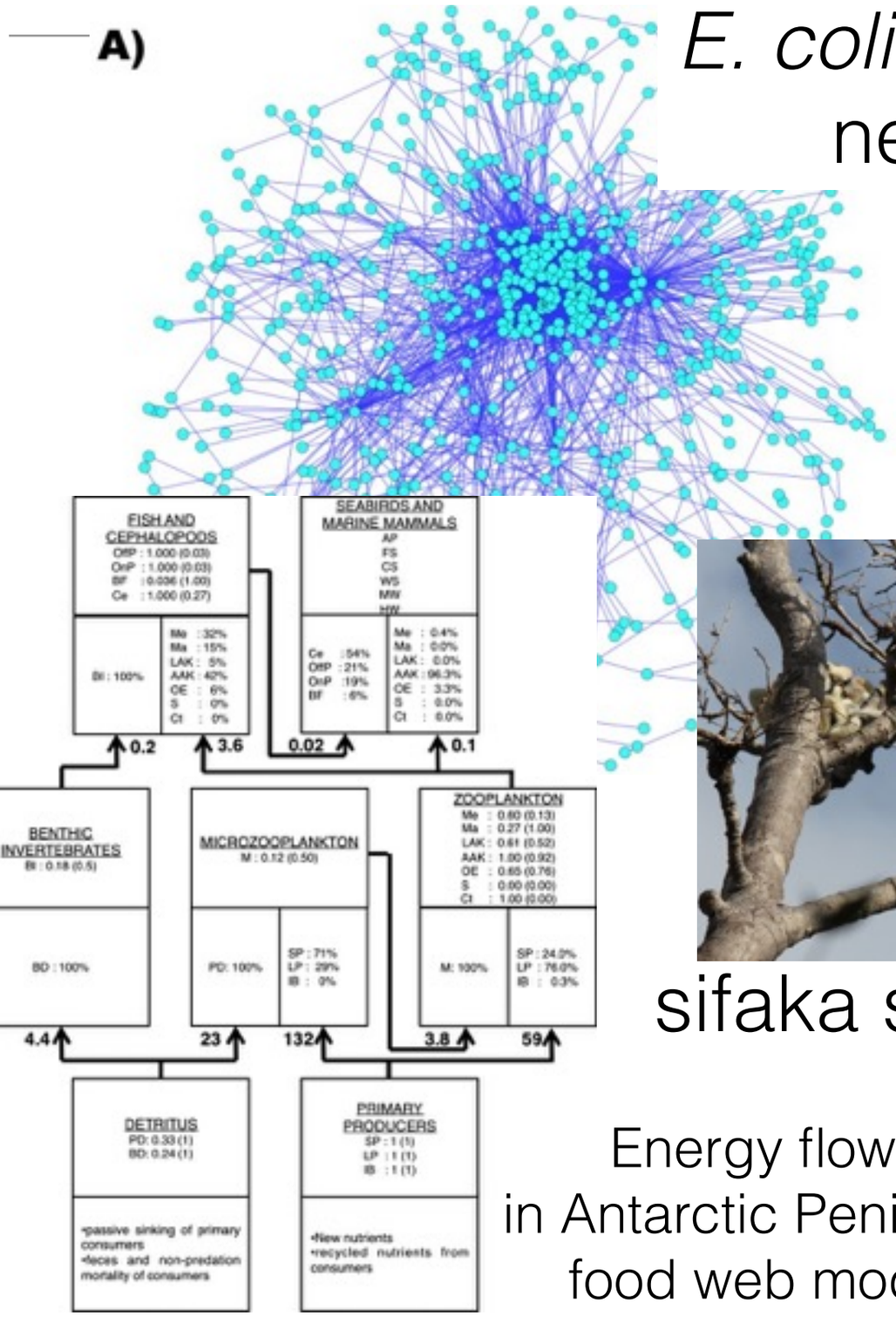
Why networks?

General yet powerful means of representing patterns of connections between the parts of a system

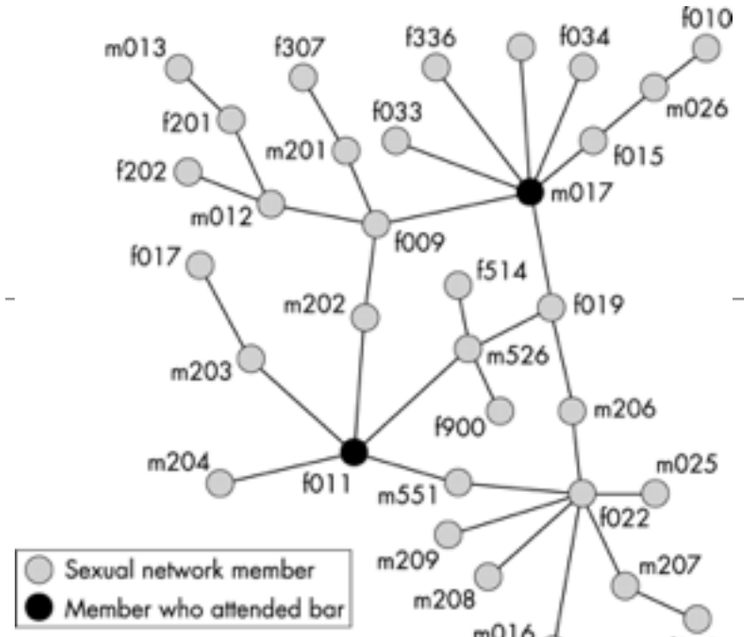
Mathematical, computational, and statistical framework for studying scientific systems:

- Statistically characterize the structure of systems
- Use models in an effort to understand how network properties arise in the first place
- Examine the interplay between structure and dynamics to predict system behavior

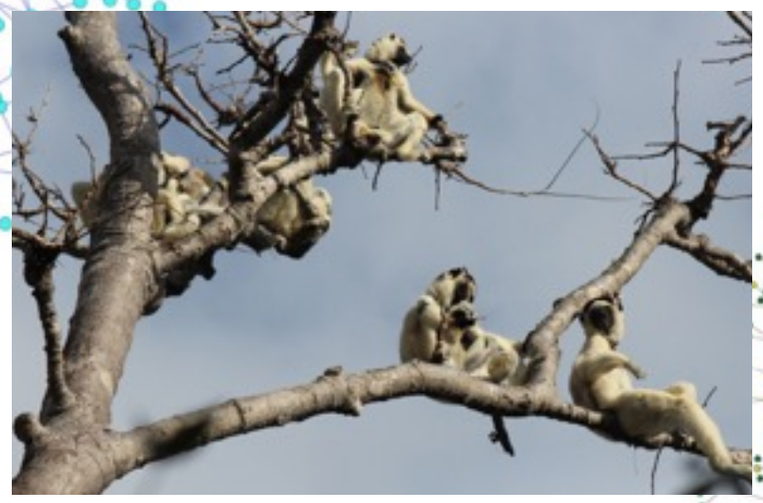
Biological Networks



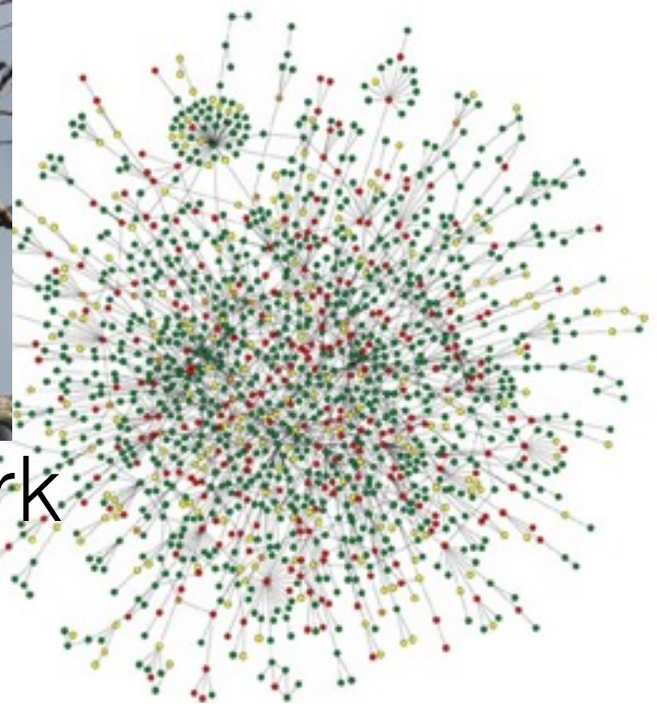
E. coli metabolic network



Alberta sexual network



sifaka social network



Energy flows in Antarctic Peninsula food web model

Saccharomyces protein-protein interaction network

Mathematics of Networks

Notation and Definitions

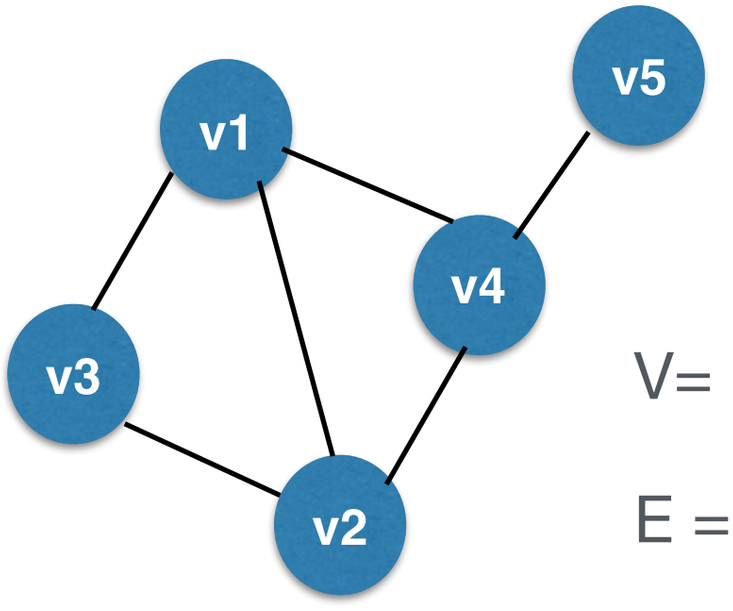
Consider an undirected network (graph) **G** with *n* vertices

G = (V, E)

V is the set of vertices

E is the set of edges

Edge **(u, v)** is the edge between vertex *u* and vertex *v*



V= {v1, v2, v3, v4, v5}

E = ({v1, v3}, {v1, v4}, {v1, v2}, {v2, v3}, {v2, v4}, {v4, v5})

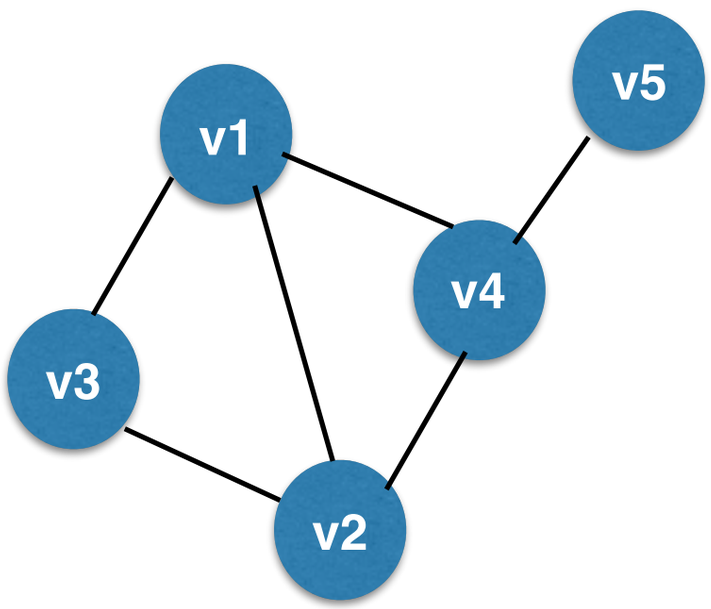
The Adjacency Matrix

One of the ways to represent a network mathematically

Each matrix entry describes a relationship between the vertices

For an undirected network with *n* vertices, the adjacency matrix is the *n x n* matrix **A** in which:

$$A_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise} \end{cases}$$

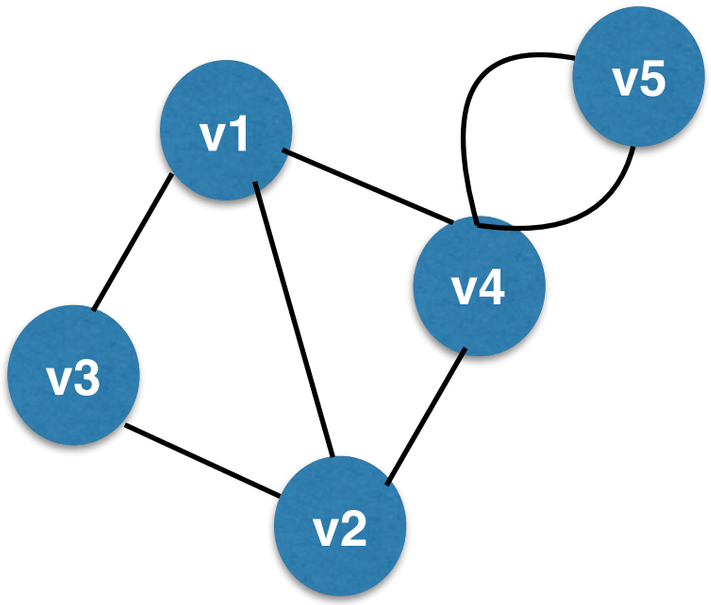


vertex *i*

vertex *j*

	v1	v2	v3	v4	v5
v1	0	1	1	1	0
v2	1	0	1	1	0
v3	1	1	0	0	0
v4	1	1	0	0	1
v5	0	0	0	1	0

Multi-edges



	v1	v2	v3	v4	v5
v1	0	1	1	1	0
v2	1	0	1	1	0
v3	1	1	0	0	0
v4	1	1	0	0	**2**
v5	0	0	0	**2**	0

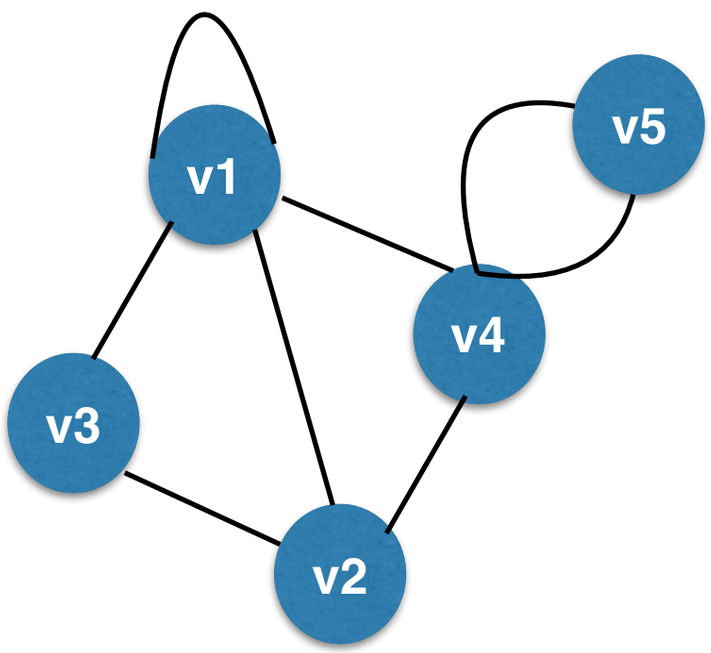
Self-edges

Set corresponding diagonal element A_{ij} to 2

Why 2 and not 1???

Need to count both ends of every edge

Non self-edges appear twice in the adjacency matrix

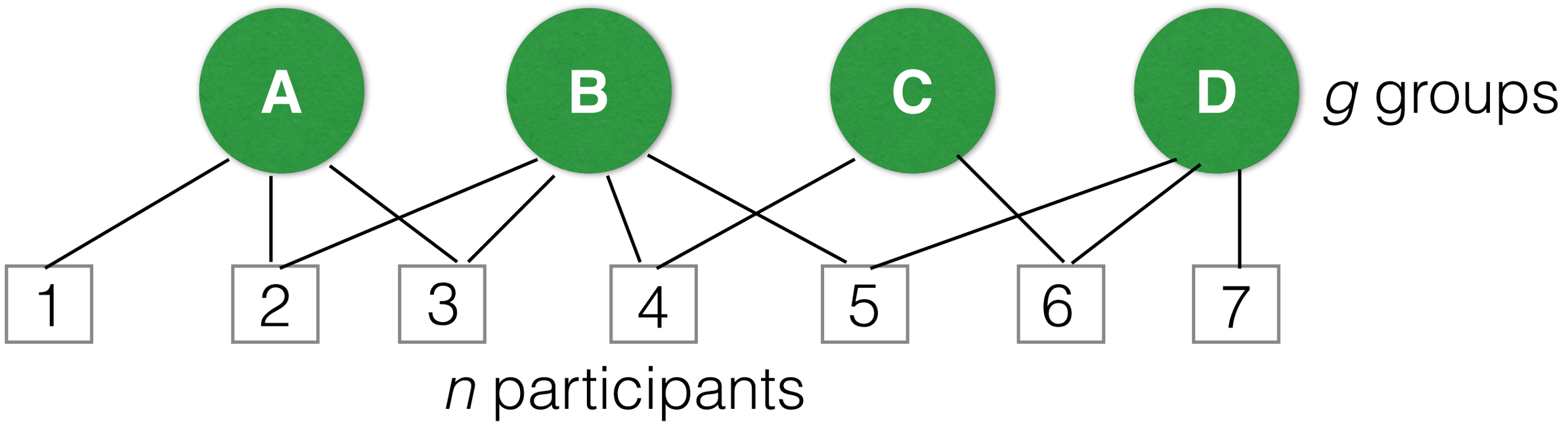


	v1	v2	v3	v4	v5
v1	**2**	1	1	1	0
v2	1	0	1	1	0
v3	1	1	0	0	0
v4	1	1	0	0	**2**
v5	0	0	0	**2**	0

Bipartite networks

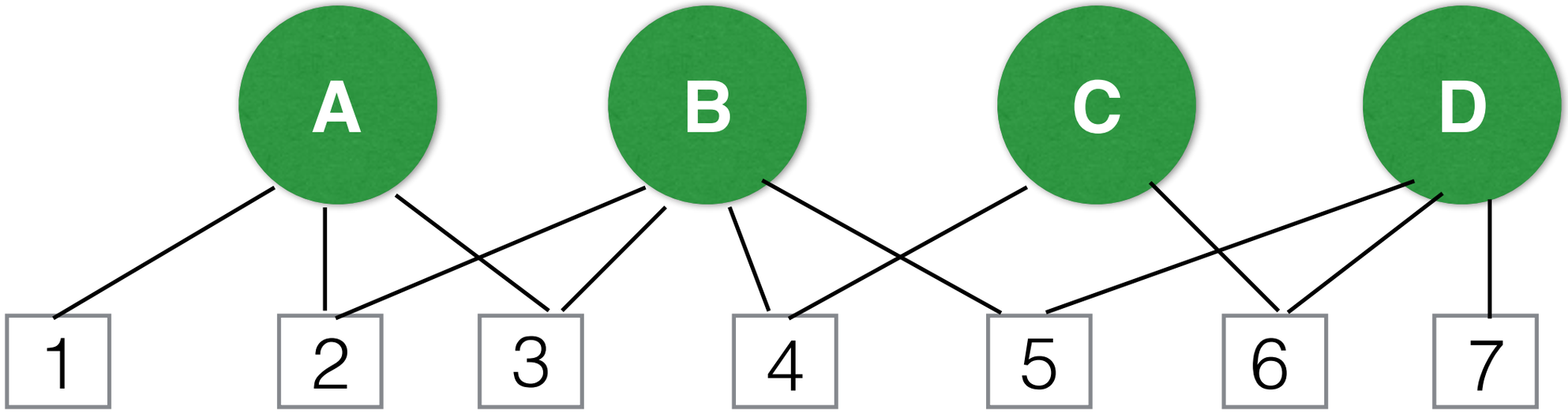
Bipartite networks contain two different types of vertices, and the edges run only between vertices of unlike types.

Examples: group membership, actor-film, author-paper, metabolites-chemical rxns



Bipartite networks

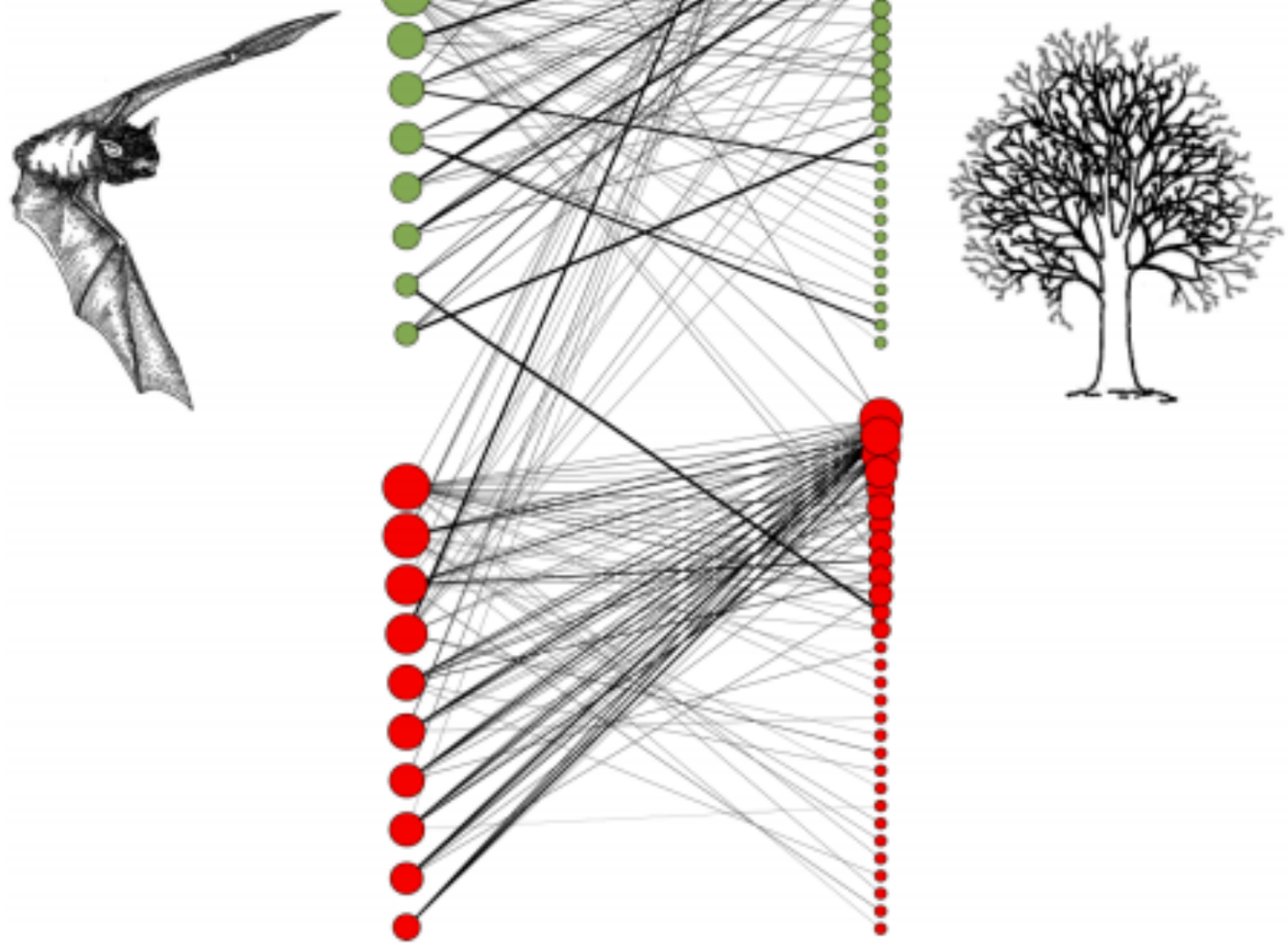
The **incidence matrix *B*** for a bipartite network is a g x n matrix with elements Bij:



$$B_{ij} = \begin{cases} 1 \text{ if participant } j \text{ belongs to } i \\ 0 \text{ otherwise} \end{cases}$$

	1	2	3	4	5	6	7
A	1	1	1	0	0	0	0
B	0	1	1	1	1	0	0
C	0	0	0	1	0	1	0
D	0	0	0	0	1	1	1

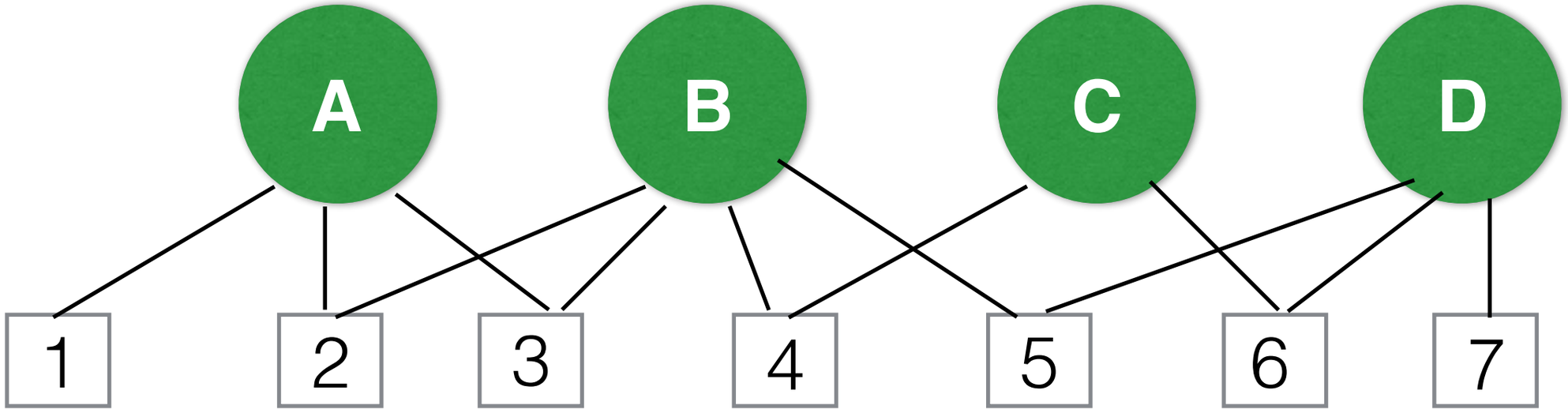
Bipartite roosting network. Vertices represent bats (left) and trees (right).



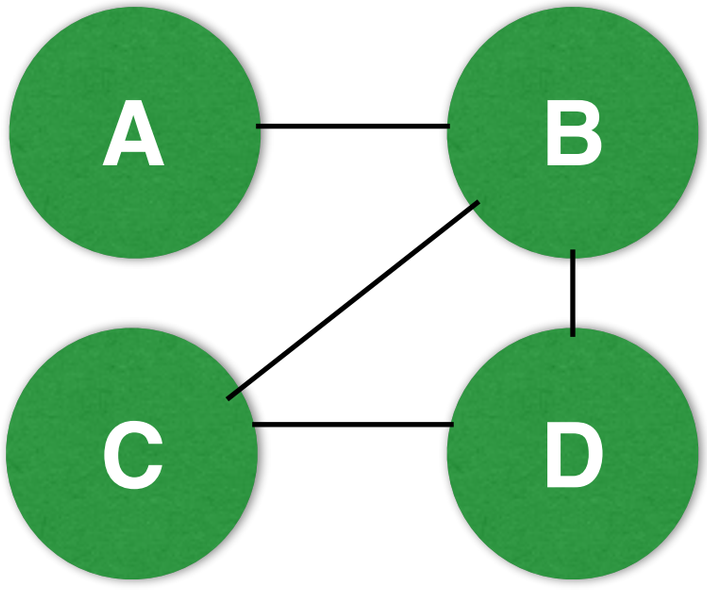
Miguel A. Fortuna, Ana G. Popa-Lisseanu, Carlos Ibáñez, and Jordi Bascompte 2009. The roosting spatial network of a bird-predator bat. Ecology 90:934–944. http://dx.doi.org/10.1890/08-0174.1

One-mode projections

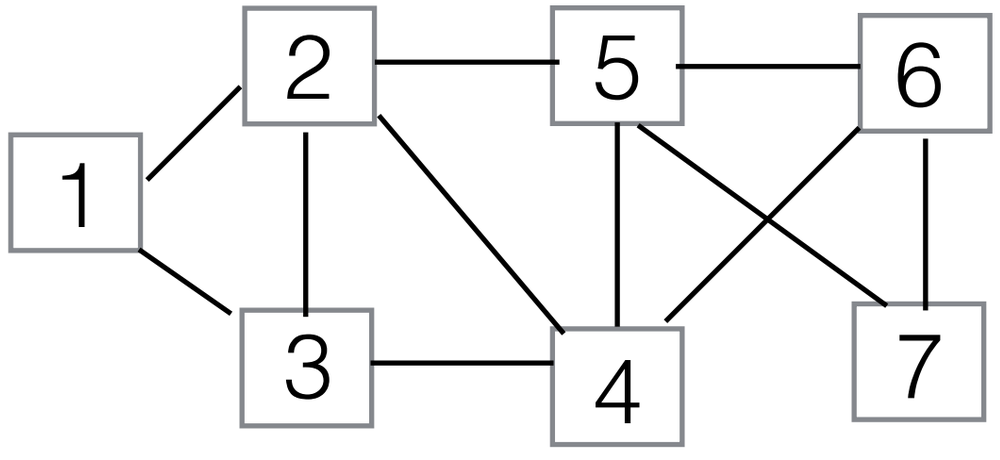
If we want to work with direct connections of vertices of just one type:



Shared participants



Common membership



Adjacency matrix from the incidence matrix

B=

	1	2	3	4	5	6	7
A	1	1	1	0	0	0	0
B	0	1	1	1	1	0	0
C	0	0	0	1	0	1	0
D	0	0	0	0	1	1	1

B^T=

	A	B	C	D
1	1	0	0	0
2	1	1	0	0
3	1	1	0	0
4	0	1	1	0
5	0	1	0	1
6	0	0	1	1
7	0	0	0	1

$$P = B^T B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 1 & 0 & 0 \\ 1 & 2 & 2 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Multiply incidence matrix by its transpose

Adjacency matrix from the incidence matrix

$$P = B^T B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 1 & 0 & 0 \\ 1 & 2 & 2 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

set diagonal to zero

set non-zero items to 1

$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

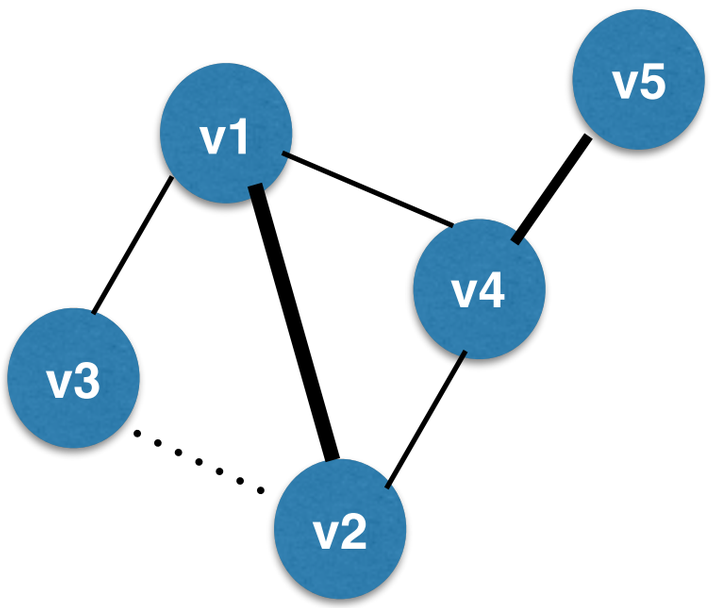
unweighted

Weighted Networks

Undirected networks have edges that form simple presence/absence connections between vertices

However, in some situations, it is useful to represent edges as having a strength, weight, or value (*e.g.*, frequency of social interaction)

Values can be positive or negative



	v1	v2	v3	v4	v5
v1	0	**4**	1	1	0
v2	**4**	0	**-0.5**	1	0
v3	1	**-0.5**	0	0	0
v4	1	1	0	0	**2**
v5	0	0	0	**2**	0

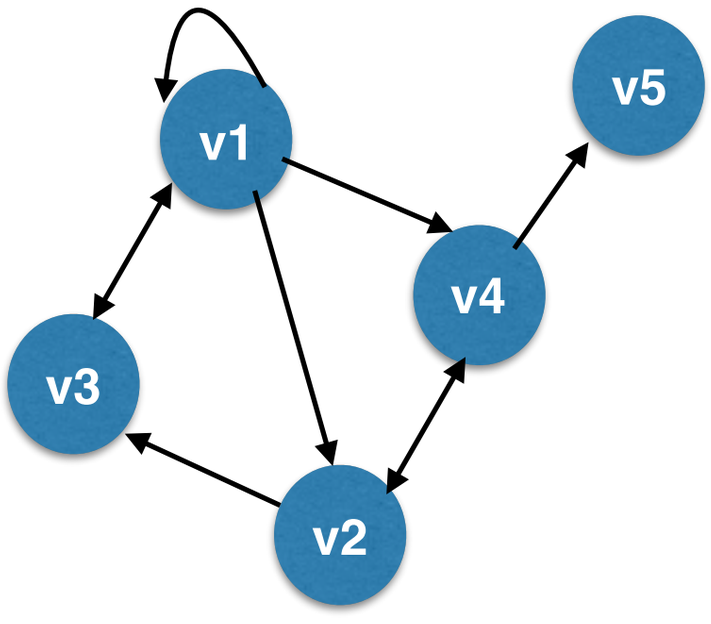
Directed Networks (Digraphs)

Networks in which each edge has a direction, pointing from an origin vertex to a destination vertex

Self-edges are given a value of 1

Example: ‘v1 groomed v2’

$$A_{ij} = \begin{cases} 1 \text{ if there's an edge from } j \text{ to } i \\ 0 \text{ otherwise} \end{cases}$$



vertex *j*

vertex *i*

	v1	v2	v3	v4	v5
v1	1	0	1	0	0
v2	1	0	0	1	0
v3	1	1	0	0	0
v4	1	1	0	0	0
v5	0	0	0	1	0

asymmetric

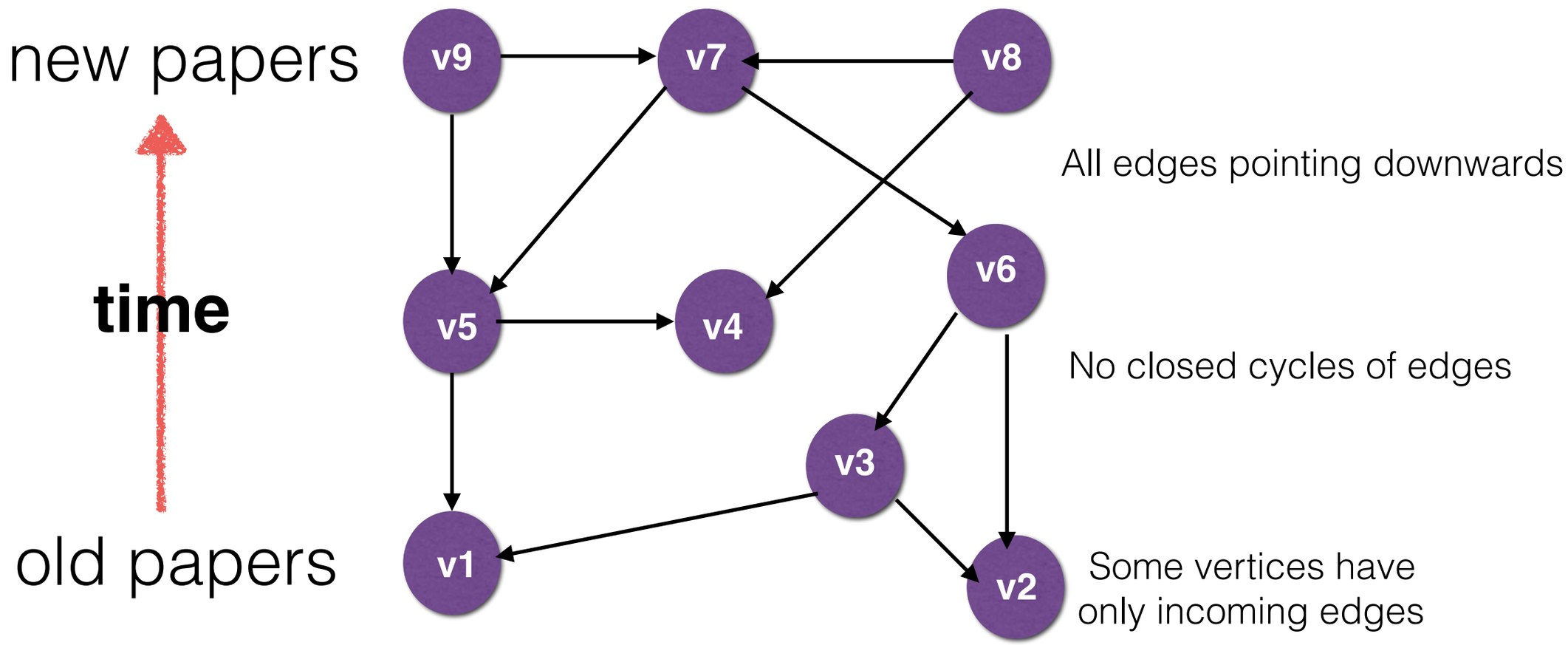
Directed Acyclic Graphs (DAGs)

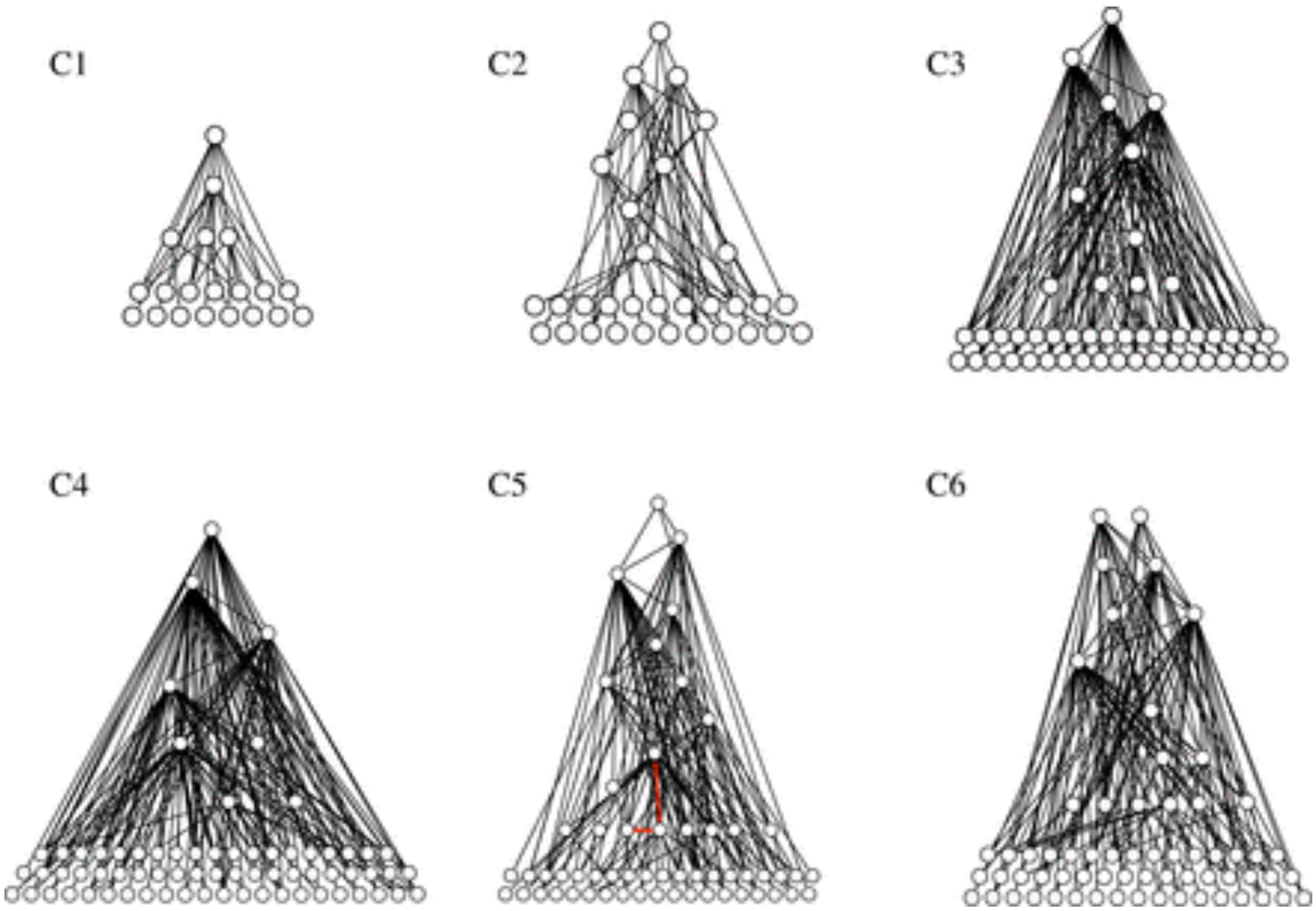
A **cycle** is a path that starts and ends at the same vertex.

Acyclic directed networks have no cycles (i.e., there is no closed loop of edges with the arrow on each of the edges pointing the same way around the loop).

A self-edge counts as a cycle; therefore, acyclic networks have no self-edges.

Examples: network of citations between papers, gene ontology





Observed dominance hierarchies in ant networks (approximate DAGs). Workers are aligned by rank.

Shimoji, Hiroyuki, et al. "Global network structure of dominance hierarchy of ant workers." Journal of The Royal Society Interface 11.99 (2014): 20140599.

igraph: Network Analysis and Visualization

http://igraph.org

“Routines for simple graphs and network analysis. It can handle large graphs very well and provides functions for generating random and regular graphs, graph visualization, centrality methods and much more.”

Vertex and edge IDs

- Vertices are always numbered from 1
- Numbering is continual, from 1 to |V|

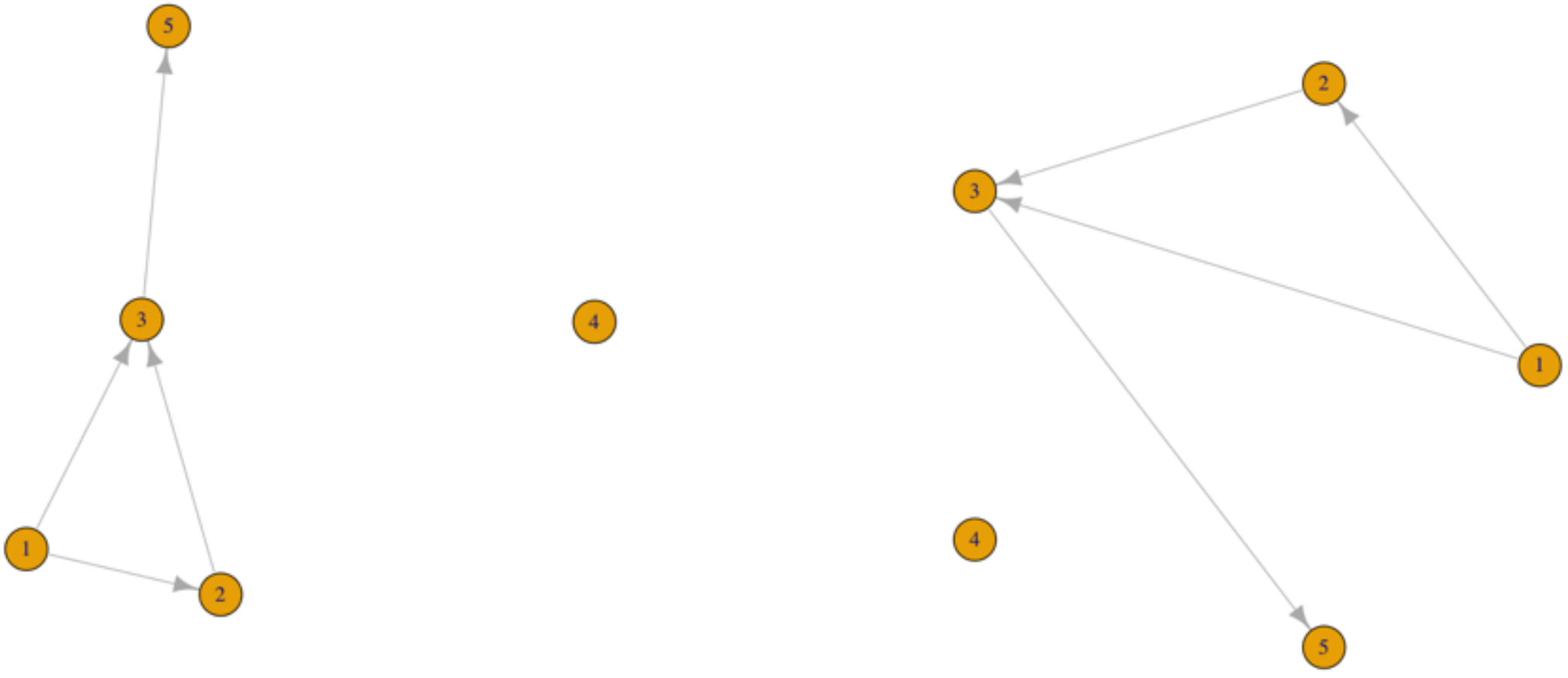
```
> g <- graph(c(1,2,1,3,2,3,3,5), n=5 )
> g
IGRAPH D--- 5 4 --
+ edges:
[1] 1->2 1->3 2->3 3->5
> summary(g)
IGRAPH D--- 5 4 --
> is.igraph(g)
[1] TRUE
> vcount(g)
[1] 5
> ecount(g)
[1] 4
> V(g) #vertex sequence
+ 5/5 vertices:
[1] 1 2 3 4 5
> E(g) #edge sequence
+ 4/4 edges:
[1] 1->2 1->3 2->3 3->5
```

Naming vertices

```
> V(g)$name <- sample(letters,vcount(g))
> V(g)$name
[1] "a" "b" "i" "e" "y"
> names(V(g))
[1] "a" "b" "i" "e" "y"
```

Visualization (layout is often arbitrary)

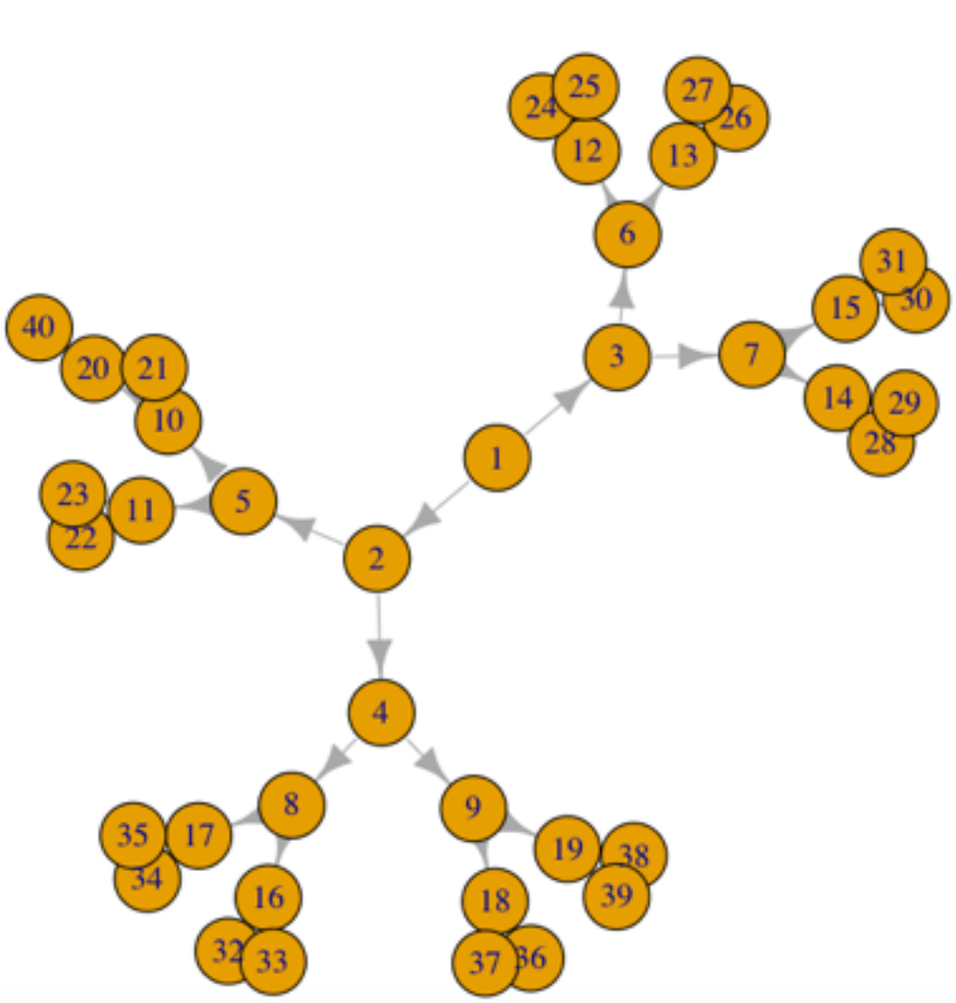
```
> plot(g)
> plot(g, layout=layout.circle)
```

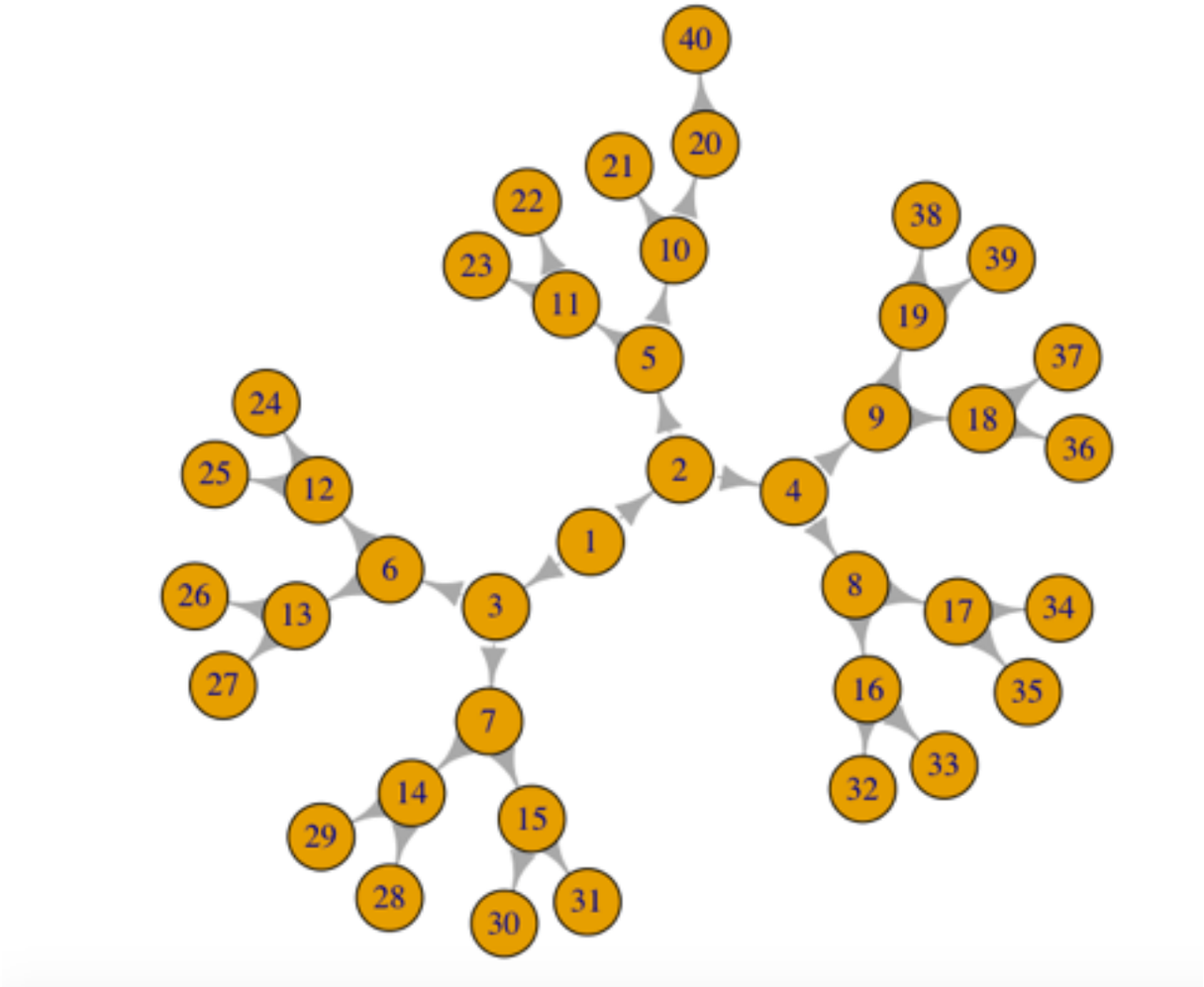


No meaning attached to edge length

Force directed layouts

```
> g<- make_tree(n=40, children=2, mode="out")
> plot(g, layout=layout.fruchterman.reingold)
> plot(g, layout=layout.kamada.kawai)
>
```





Example dataset: animal social network

- A social network is any number of individuals interconnected via social ties (sexual, cooperative, etc.) between them
- In nearly all animal social networks, each vertex is an individual
- Properties of individuals are vertex attributes
- Pairs of vertices form a dyad when there is an edge between them

```
> contacts <- read.csv("network_intro.csv", header=TRUE)
> names(contacts) #column names
[1] "Initiator" "Receiver"  "Init.Sex"  "Rec.Sex"   "Behavior"
> head(contacts)
  Initiator Receiver Init.Sex Rec.Sex Behavior
1        Qu       Gl        M       M        G
2        He       Va        F       F       AC
3        Py       Ro        F       F       AC
4        Py       Ro        F       F        G
5        Ro       Py        F       F       IG
6        Ca       Fa        F       F       CO
```

Create a weighted adjacency matrix

Edges can represent one or several behavior types

In this example, we're looking at frequency of grooming interactions

```
#want to look at only grooming behavior (grooming and mutual grooming)
contacts <- contacts[contacts$Behavior=="G" | contacts$Behavior=="MG",]

#need to do this so that we have a square adjacency matrix
listnames <- levels(as.factor(c(as.character(contacts$Initiator), as.character(contacts$Receiver))))
contacts$Initiator <- factor(contacts$Initiator, levels=listnames)
contacts$Receiver <- factor(contacts$Receiver, levels=listnames)

#created a weighted adjacency matrix from the edgelist
m1 <- table(contacts$Initiator, contacts$Receiver)
m1
adj1 <- as.matrix(m1)
adj1
```

Weighted adjacency matrix (frequency of contacts)

```
> adj1
```

	Ab	As	Ca	Da	Fa	Fr	Gl	He	Ki	Om	Pe	Py	Qu	Ro	Sa	Ti	Va	Vy	Xa	Za	Ze
Ab	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
As	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
Ca	0	0	0	0	2	0	0	0	0	0	3	0	0	1	0	0	0	0	0	0	0
Da	0	0	0	0	0	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Fa	0	0	0	0	0	6	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0
Fr	0	0	0	3	5	0	0	0	0	0	0	0	0	3	0	1	0	0	0	0	0
Gl	0	0	0	0	0	0	0	8	0	0	0	0	4	0	0	0	7	11	0	0	0
He	0	0	0	0	0	0	8	0	0	0	0	0	1	0	0	0	33	18	0	0	0
Ki	0	0	0	0	0	0	0	0	0	3	0	0	0	2	4	0	0	0	0	2	0
Om	0	0	0	0	0	0	0	0	2	0	0	0	0	13	0	0	0	0	0	2	0
Pe	0	0	0	0	2	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
Py	0	0	0	1	0	0	0	0	0	0	0	0	1	12	0	6	0	0	0	0	0
Qu	0	0	1	0	0	0	7	1	0	0	0	1	0	0	0	3	2	0	0	0	0
Ro	0	0	0	0	1	5	0	0	6	19	8	13	2	0	1	2	0	0	0	11	1
Sa	0	0	0	0	0	0	0	0	3	0	0	0	0	3	0	0	0	0	0	2	0
Ti	1	4	0	0	3	2	0	0	0	0	2	1	7	0	0	0	0	0	3	0	0
Va	0	0	0	0	0	0	2	3	0	0	0	0	1	0	0	0	0	17	0	0	0
Vy	0	0	0	0	0	0	1	8	0	0	0	0	0	0	0	0	6	0	0	0	0
Xa	8	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Za	0	0	0	0	0	0	0	0	4	0	0	0	0	9	15	0	0	0	0	0	1
Ze	0	0	0	0	0	0	0	0	0	6	0	0	0	2	2	0	0	0	0	0	0

```
> lemur.graph <- graph_from_adjacency_matrix(adj1, mode="directed", weighted=TRUE, diag=FALSE, add.colnames=NULL)
> # diag = FALSE (diagonal is zero'd out; no self edges)
> # add.colnames = NULL (if present, column names are added as vertex attribute name)
>
> summary(lemur.graph)
IGRAPH DNW- 21 80 --
+ attr: name (v/c), weight (e/n)
> #D- Directed; N-Named; W-Weighted; U-Unweighted; B- Bipartite (if the 'type' vertex attribute is set)
> # number of vertices then number of edges
> # Vertex and edge attributes
>
> print.igraph(lemur.graph) # if you want to see the edges of the graph also
IGRAPH DNW- 21 80 --
+ attr: name (v/c), weight (e/n)
+ edges (vertex names):
 [1] Ab->As Ab->Xa As->Ab As->Xa Ca->Fa Ca->Pe Ca->Ro Da->Fr Da->Pe Fa->Fr Fa->Pe Fr->Da Fr->Fa Fr->Ro Fr->Ti Gl->He Gl->Qu
[18] Gl->Va Gl->Vy He->Gl He->Qu He->Va He->Vy Ki->Om Ki->Ro Ki->Sa Ki->Za Om->Ki Om->Ro Om->Za Pe->Fa Pe->Ro Py->Da Py->Qu
[35] Py->Ro Py->Ti Qu->Ca Qu->Gl Qu->He Qu->Py Qu->Ti Qu->Va Ro->Fa Ro->Fr Ro->Ki Ro->Om Ro->Pe Ro->Py Ro->Qu Ro->Sa Ro->Ti
[52] Ro->Za Ro->Ze Sa->Ki Sa->Ro Sa->Za Ti->Ab Ti->As Ti->Fa Ti->Fr Ti->Pe Ti->Py Ti->Qu Ti->Xa Va->Gl Va->He Va->Qu Va->Vy
[69] Vy->Gl Vy->He Vy->Va Xa->Ab Xa->As Za->Ki Za->Ro Za->Sa Za->Ze Ze->Om Ze->Ro Ze->Sa
```

Edge list

Not very helpful

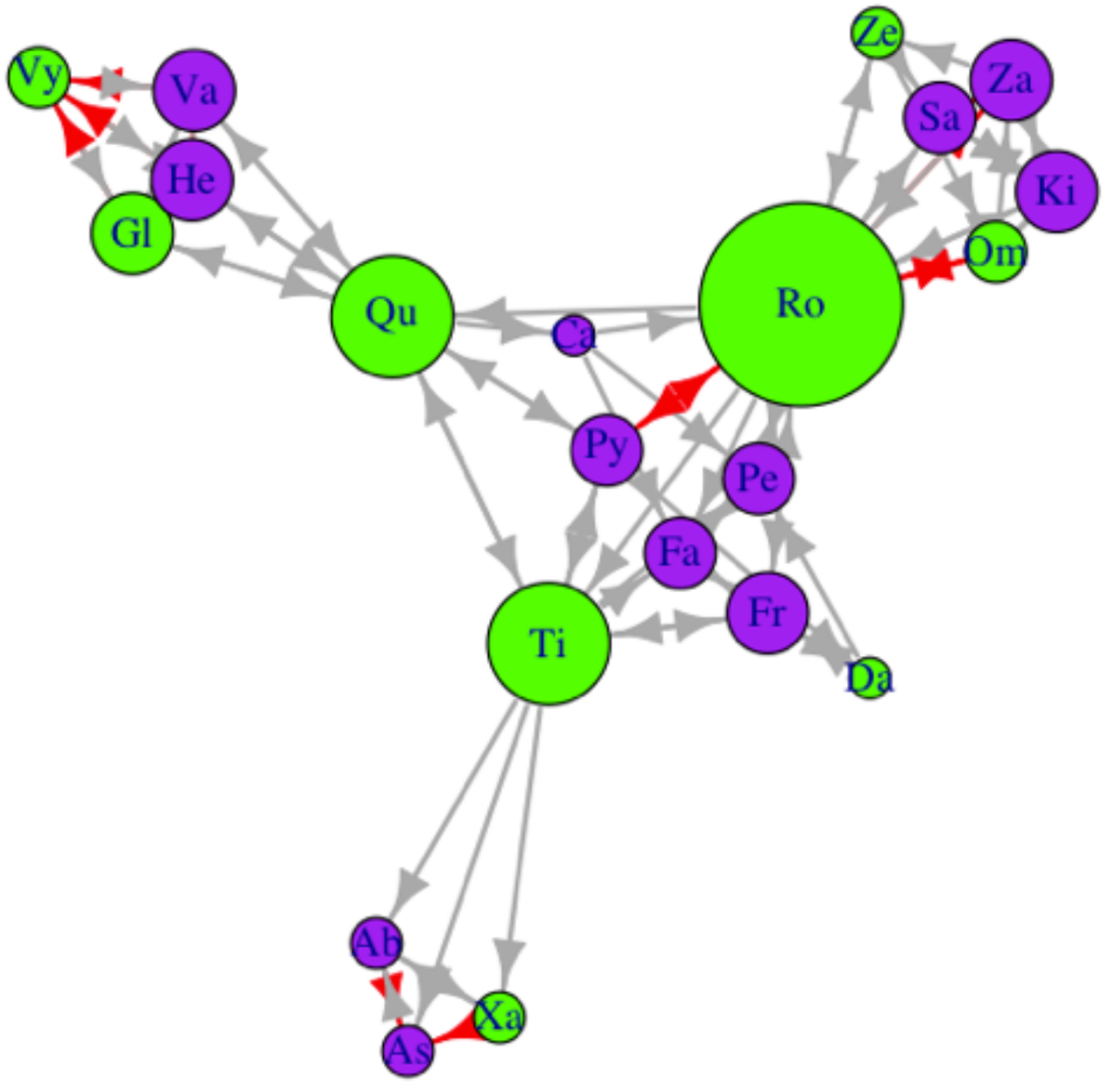
```
> get.edgelist(lemur.graph)
      [,1] [,2]
 [1,] "Ab" "As"
 [2,] "Ab" "Xa"
 [3,] "As" "Ab"
 [4,] "As" "Xa"
 [5,] "Ca" "Fa"
 [6,] "Ca" "Pe"
 [7,] "Ca" "Ro"
 [8,] "Da" "Fr"
 [9,] "Da" "Pe"
[10,] "Fa" "Fr"
[11,] "Fa" "Pe"
[12,] "Fr" "Da"
[13,] "Fr" "Fa"
[14,] "Fr" "Ro"
[15,] "Fr" "Ti"
[16,] "Gl" "He"
[17,] "Gl" "Qu"
[18,] "Gl" "Va"
[19,] "Gl" "Vy"
[20,] "He" "Gl"
```

Assigning attributes to vertices

```
> # assign 'sex' attribute
> x <- as.factor(c(as.character(contacts$Initiator),as.character(contacts$Receiver)))
> y <- as.factor(c(as.character(contacts$Init.Sex),as.character(contacts$Rec.Sex)))
> tt <- table(x,y)
> sex <- 1*(tt[,2]>0)
> sex #assign a 0 or 1 for sex
Ab As Ca Da Fa Fr Gl He Ki Om Pe Py Qu Ro Sa Ti Va Vy Xa Za Ze
 0  0  0  1  0  0  1  0  0  1  0  0  1  1  0  1  0  1  1  0  1
>
> # change to male "M" and female "F"
> n <- length(sex)
> for (i in 1:n){
+   if(sex[i]=="0"){
+           sex[i]="F"
+       } else if (sex[i]=="1"){
+           sex[i]="M"
+       }
+ }
> V(lemur.graph)$sex <- sex
> V(lemur.graph)$sex
 [1] "F" "F" "F" "M" "F" "F" "M" "F" "F" "M" "F" "F" "M" "M" "F" "M" "F" "M" "M" "F" "M"
> summary(lemur.graph)
IGRAPH DNW- 21 80 --
+ attr: name (v/c), sex (v/c), weight (e/n)
```

Visualization

```
> plot(lemur.graph, layout=lay, vertex.label=V(lemur.graph)$name, vertex.color=ifelse(sex=="M","green","purple"),
vertex.size=nodesize)
> nodesize <- degree(lemur.graph) * 2
> plot(lemur.graph, layout=lay, vertex.label=V(lemur.graph)$name, vertex.color=ifelse(sex=="M","green","purple"),
vertex.size=nodesize)
> #'fancier' plotting
> nodesize <- degree(lemur.graph) * 2
> lay <- layout.fruchterman.reingold(lemur.graph)
> E(lemur.graph)$width <- 2
> E(lemur.graph)[E(lemur.graph)$weight > 10]$color
> E(lemur.graph)[E(lemur.graph)$weight < 10]$color
> plot(lemur.graph, layout=lay, vertex.label=V(lem
vertex.size=nodesize)
```



Vertex Centrality Measures

Vertex Centrality Measures

- Centrality measures are used to differentiate the importance or influence of individuals in the network
  - What is the most important protein in a metabolic network?
  - Which individuals should we target for vaccination?
  - Which are the keystone species in an ecosystem?

Centrality Rules

- We cannot compare centrality measures for different networks.

- We cannot compare different kinds of centrality measures on the same network.

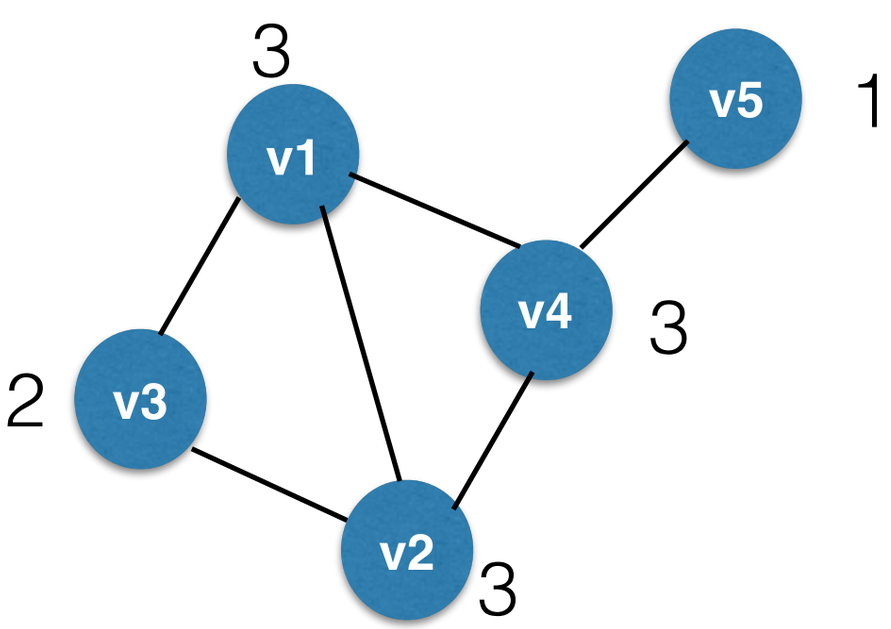
Degree Centrality

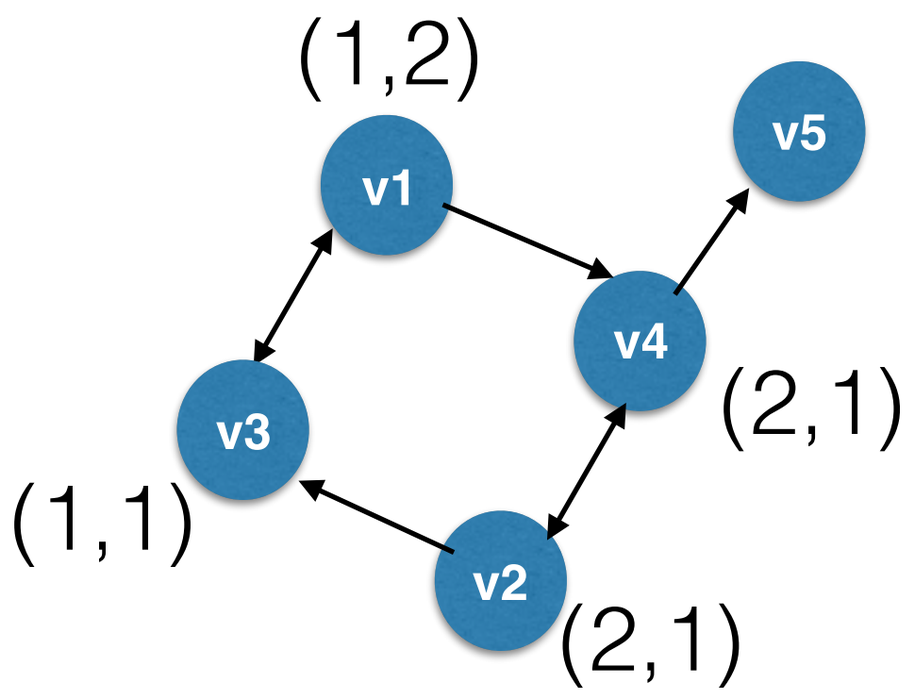
Assumption: The most connected vertices are the most central.

- **Degree** is the number of edges connected to a vertex.

- In directed networks, vertices have an **in-degree** and an **out-degree** (the number of edges arriving and leaving, respectively).

- **Weighted degree (node strength)** is the sum of the edge weights connected to a vertex





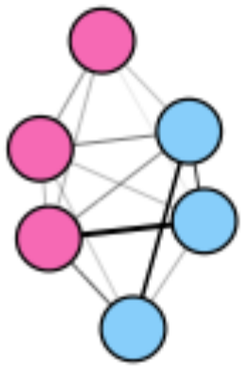
Path-following centrality measures

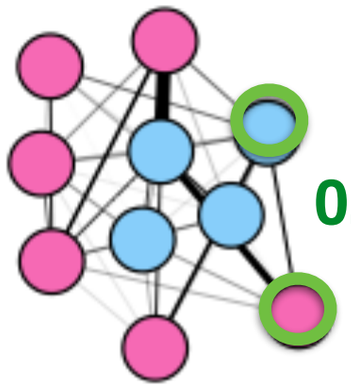
Betweenness, closeness, informational centrality, etc. count paths (between pairs of vertices) that pass through the vertex of interest

A **shortest path** (or **geodesic**) between two vertices is the minimum number of edges you have to travel across to move from one vertex to another. There may be multiple different geodesics, all of the same length.

The length of a shortest path between (*u, v*) is called the **geodesic distance** or **graph distance**.

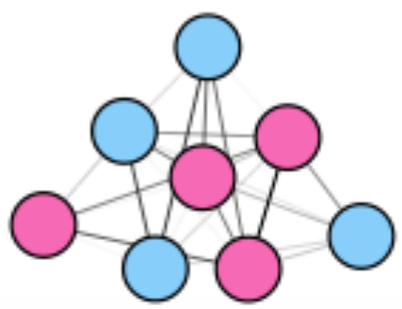
Shortest path: min. number of edges between two individuals
Weighted shortest path: if edge weight is strength of interaction, add up inverse edge weights





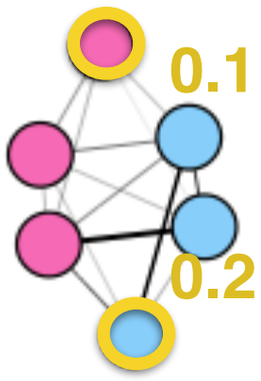
Unweighted
Shortest path = 1

Weighted
Shortest path = 10

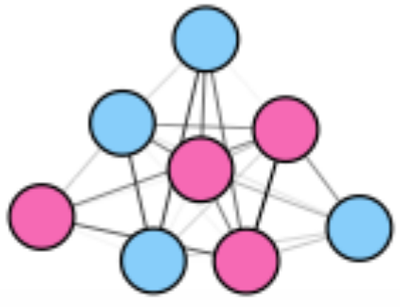


Shortest path: min. number of edges between two nodes
Weighted shortest path: add up inverse edge weights

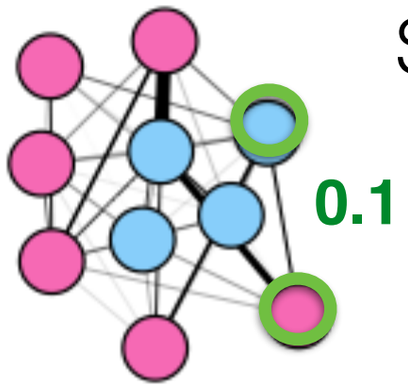
Unweighted
Shortest path = 2



Weighted
Shortest path = 15

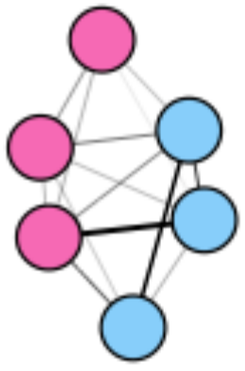


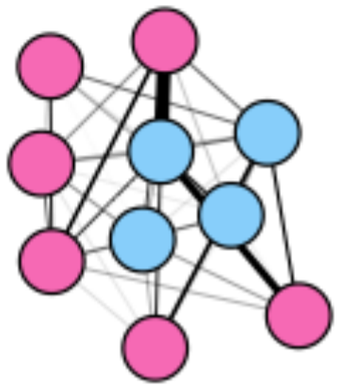
Unweighted
Shortest path = 1



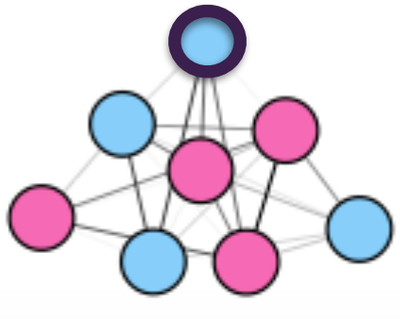
Weighted
Shortest path = 10

Shortest path: min. number of edges between two nodes (“distance”)
Weighted shortest path: add up inverse edge weights





Disconnected
Shortest path = ∞



Closeness Centrality

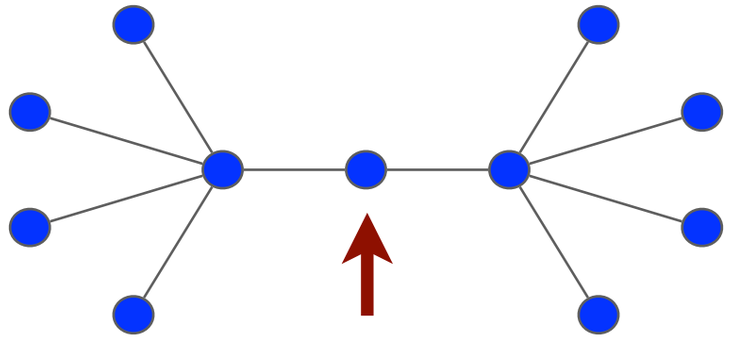
Closeness centrality measures the mean distance from a vertex to other vertices.

$$C_i = \frac{1}{l_i} = \frac{n}{\sum_j d_{ij}}$$

number of vertices divided by sum of geodesic distances from *i* to *j*

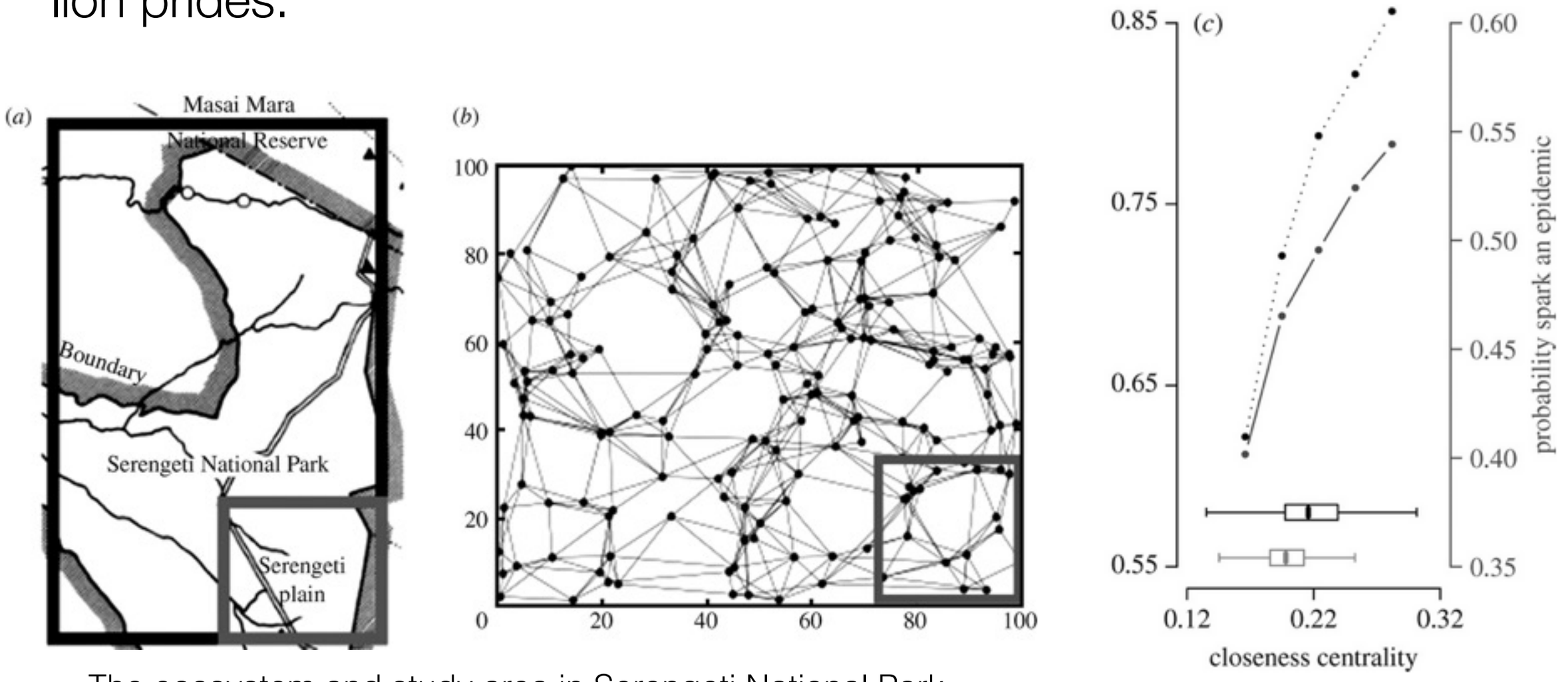
inverse of mean geodesic distance

Vertices close to other vertices are important.



Closeness Centrality Example

Epidemiological risk correlates with closeness centrality for Serengeti lion prides.



The ecosystem and study area in Serengeti National Park (left) and a simulated lion population based on estimates of territory locations and adjacencies from Serengeti Lion Project data (right).

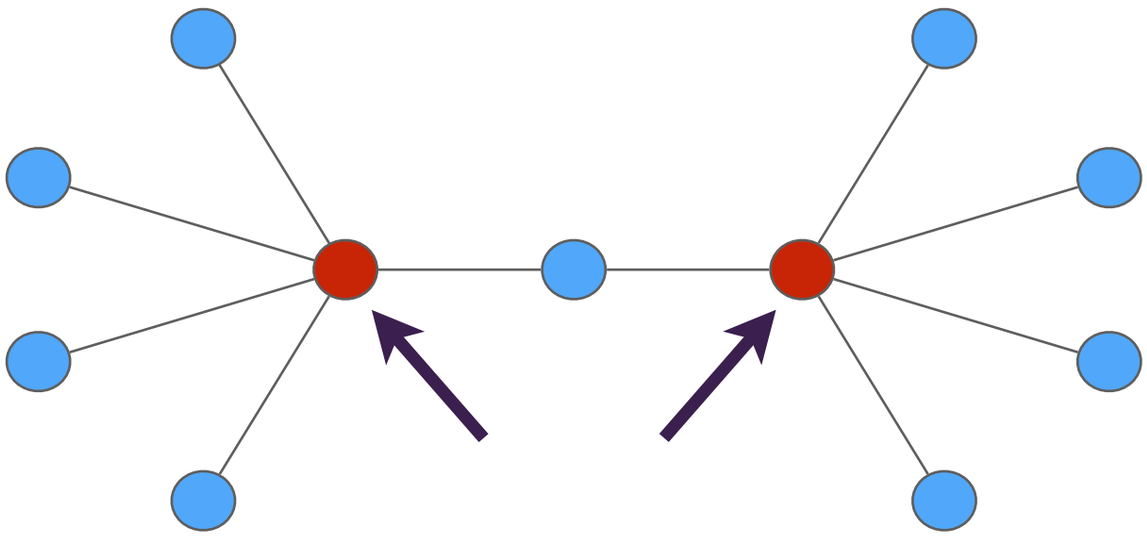
Craft, Meggan E., et al. "Distinguishing epidemic waves from disease spillover in a wildlife population." Proceedings of the Royal Society B: Biological Sciences (2009): rspb-2008.

Betweenness Centrality

Betweenness centrality: measures the extent to which a vertex lies on shortest paths between other vertices.

Vertices that lie on the shortest paths to other nodes are important because they control “information” passing between other individuals.

The removal of vertices with high betweenness causes the most disruption.



σ_{ij} number of shortest paths between vertices i and j

$\sigma_{ij}(v)$ number of those paths that pass through vertex v

$$\delta_{ij}(v) = \sigma_{ij}(v)/\sigma_{ij}$$

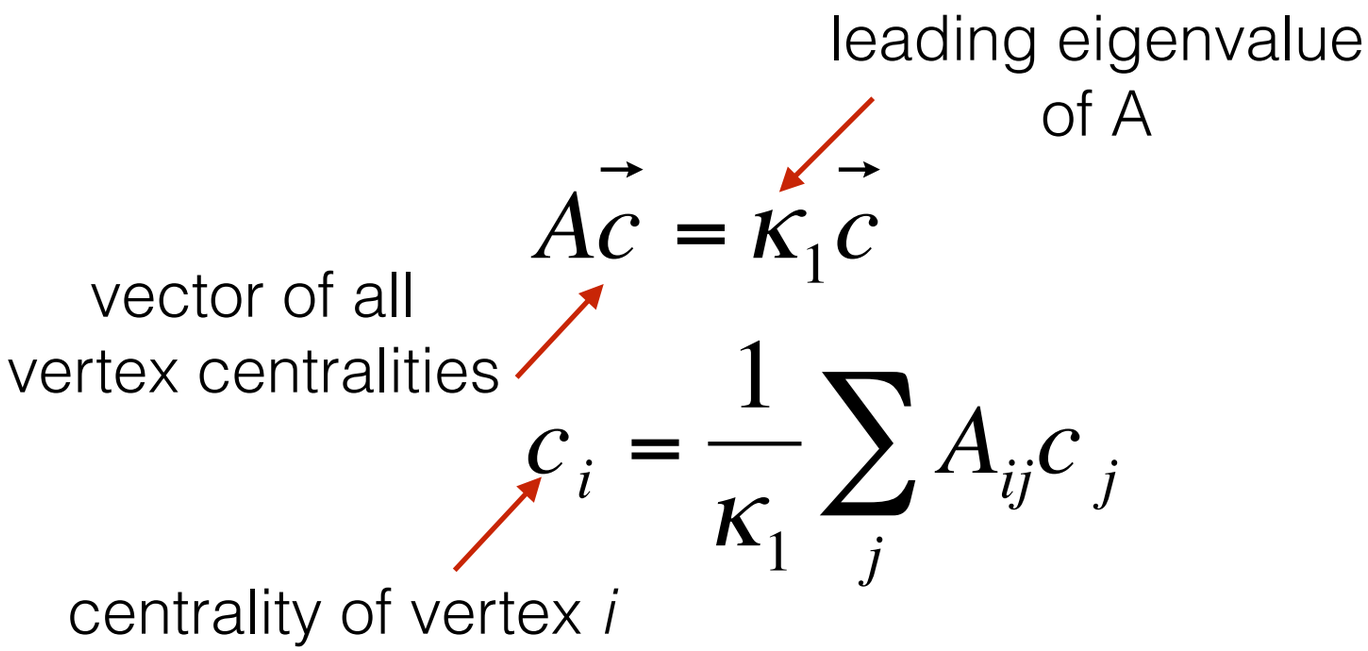
$$C(v) = \sum_{i \neq v} \sum_{j > i, j \neq v} \delta_{ij}(v)$$

Eigenvector Centrality

A vertex's importance can be increased by having connections to other vertices that are *themselves* important.

The **eigenvector centrality** of a vertex is proportional to the sum of the eigenvector centralities of its neighbors.

Works best in the case of undirected networks.



Eigenvector Centrality Issues

A directed network has an asymmetric adjacency matrix, and thus has two sets of eigenvectors (and two leading eigenvalues). Typically use the right eigenvectors- represents other vertices pointing towards each vertex.

Vertices with only out-degree have centrality zero.

Only vertices in strongly connected components can have non-zero eigenvector centrality.

Solutions: **Katz centrality** (each vertex gets a small amount of centrality “for free”), **PageRank centrality** (variation of Katz; centrality derived from neighbors is proportional to their centrality *divided by their out-degree*).

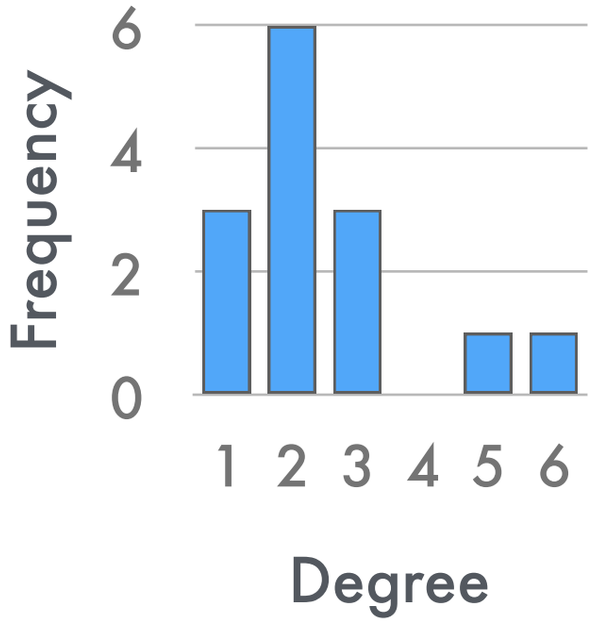
Network-level measures of structure

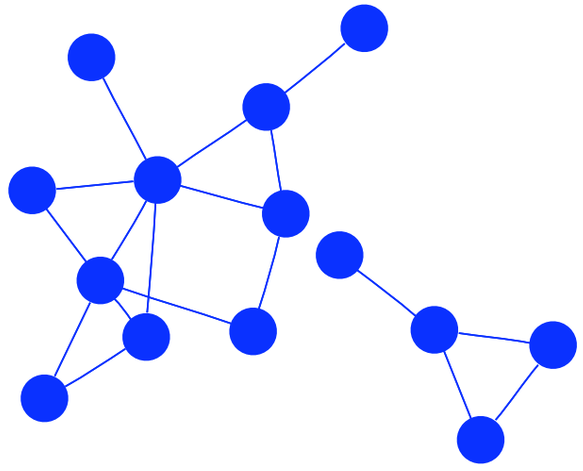
Degree Distribution

The **degree distribution** of a network is the number or fraction of vertices with each possible degree.

p_k = fraction of nodes in the network with degree k

p_k is also the probability that a randomly chosen node has degree k





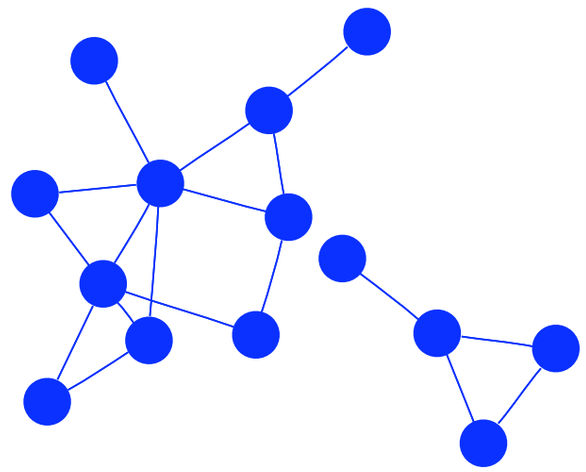
Degree sequence:
(1,1,1,2,2,2,2,2,3,3,3,3,5,6)

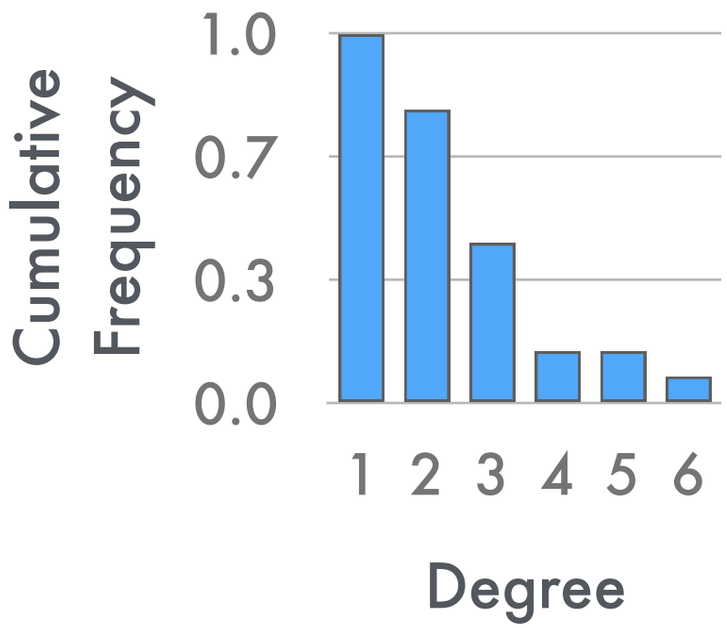
Degree	Number of nodes	Fraction of nodes
1	3	0.21
2	6	0.43
3	3	0.21
4	0	0.00
5	1	0.07
6	1	0.07
Total	14	1

Cumulative Degree Distribution

The **cumulative degree distribution** P_k gives the fraction of vertices with degree greater than or equal to k.

P_k is also the probability that a randomly chosen vertex has degree <u>at least</u> k





Degree	Number of nodes	Fraction of nodes	Cumulative Frequency
1	3	0.21	1.00
2	6	0.43	0.79
3	3	0.21	0.43
4	0	0.00	0.14
5	1	0.07	0.14
6	1	0.07	0.07
Total	14	1	

Mean degree

What is the relationship between the sum of degrees and the number of edges in the graph?

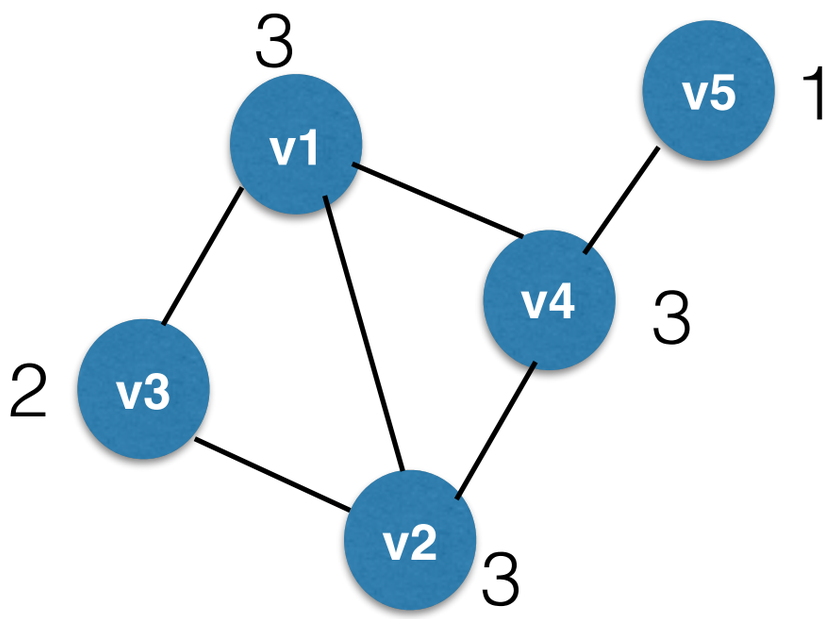
m: number of edges

n: number of vertices

$$2m = \sum_{i=1}^{n} k_i \qquad m = \left(\frac{1}{2}\right)\sum_{i=1}^{n} k_i = \left(\frac{1}{2}\right)\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}$$

k_i: degree of vertex *i*

What is the average degree (*c*) of the network?



$$c = \frac{1}{n}\sum_{i=1}^{n} k_i \qquad = \left(\frac{1}{5}\right)(3+3+2+3+1) = 2.4$$

$$\boxed{c = \frac{2m}{n}}$$

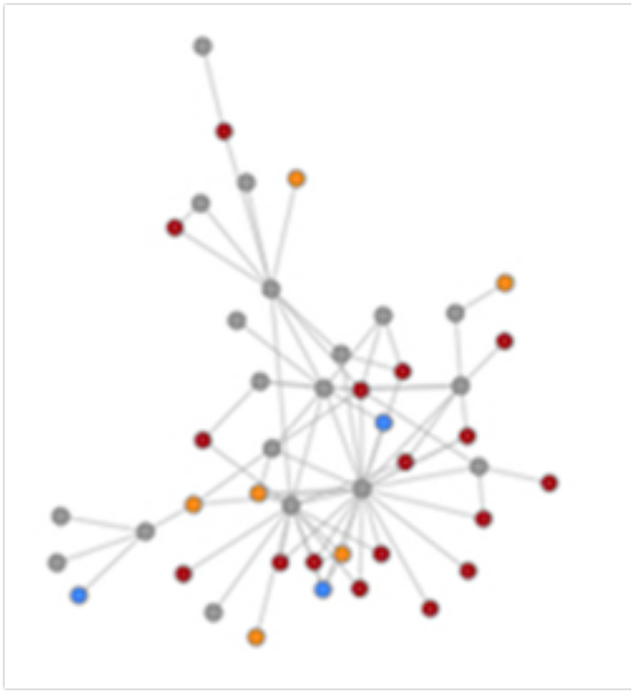
Components

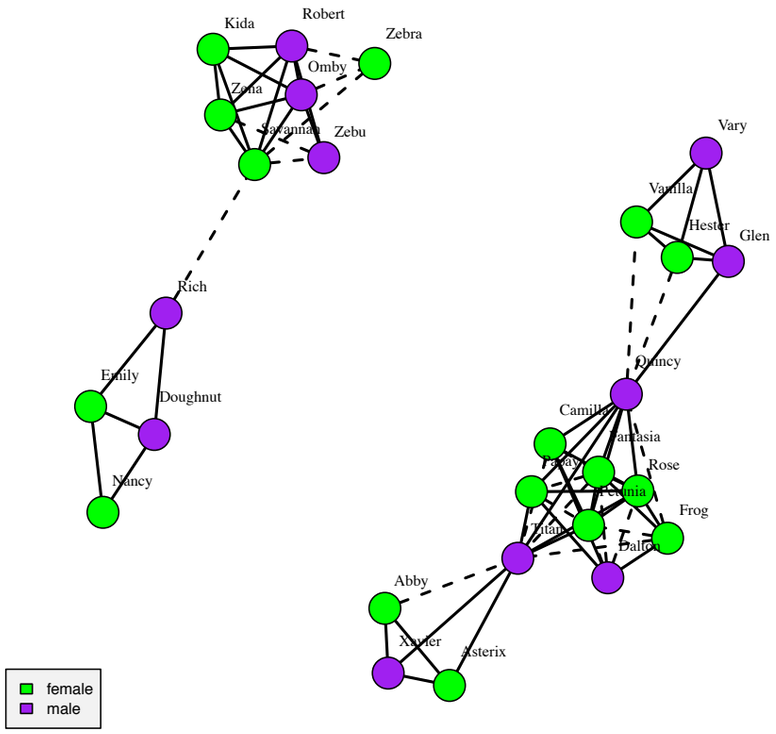
Connectivity is the number of independent paths between a pair of vertices.

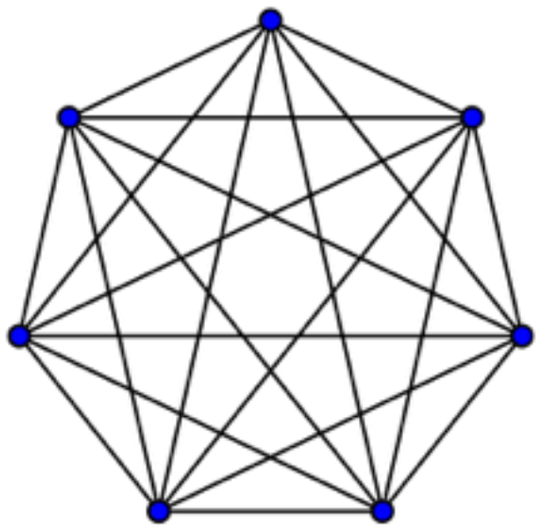
A **connected** network is one in which all pairs of vertices can be connected by a path.

It is possible for there to be no path at all between a given pair of vertices. A **disconnected network** consists of disjoint **connected components** (subgroups).

A **complete** network is one in which there are edges connecting every pair of vertices.







Graph Partitioning and Community Detection

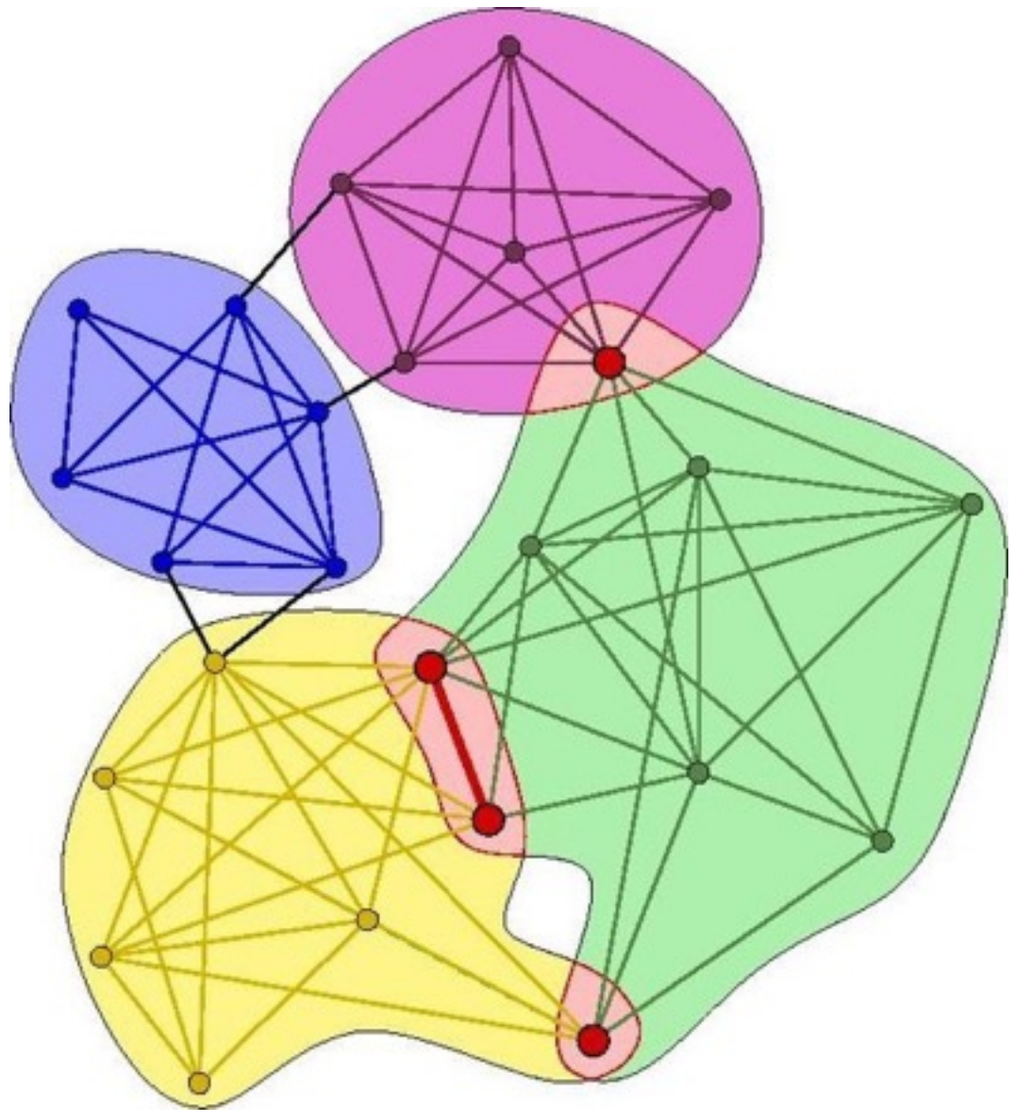
Community detection finds the natural fault lines along with a network separates. The group sizes and numbers are unspecified.

- Used as a tool to understand the structure of a network.

A network has **modularity** or **community structure** if its vertices fall into groups which have high densities of edges within them, and lower densities of edges between them.

Graph partitioning divides the vertices of a network into a given number of non-overlapping groups of given sizes such that the number of edges between groups is minimized.

- Performed as a way to divide up network into smaller and more manageable pieces.

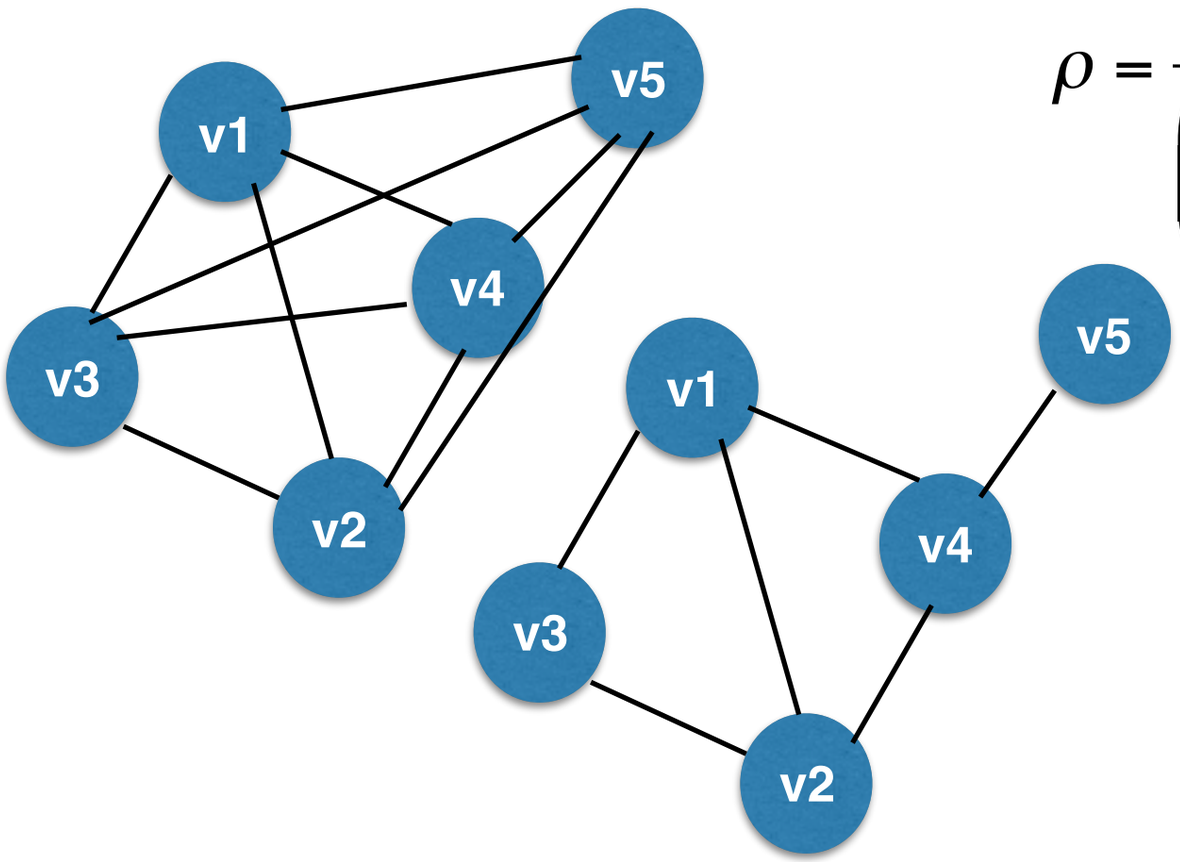


Network Density

What is the **maximum number of edges** in an undirected graph (no multi-edges or self edges)?

$$\binom{n}{2} = \frac{1}{2} n(n-1) \qquad \binom{5}{2} = \frac{1}{2} 5(5-1) = 10$$

The **density** of a graph is the fraction of all possible edges actually present.



$$\rho = \frac{m}{\binom{n}{2}} = \frac{m}{\frac{1}{2} n(n-1)} = \frac{2m}{n(n-1)} = \frac{12}{20} = 0.6$$

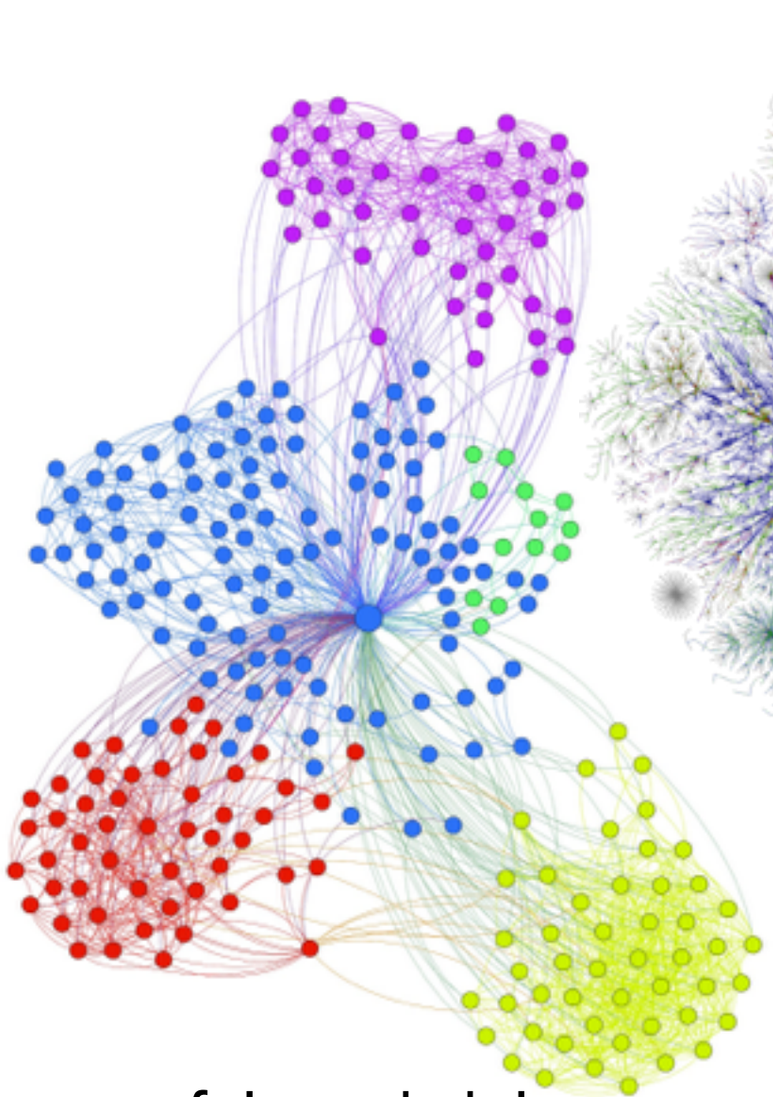
mean degree

$$\rho = \frac{2m}{n(n-1)} = \frac{c}{n-1} = 0.6$$

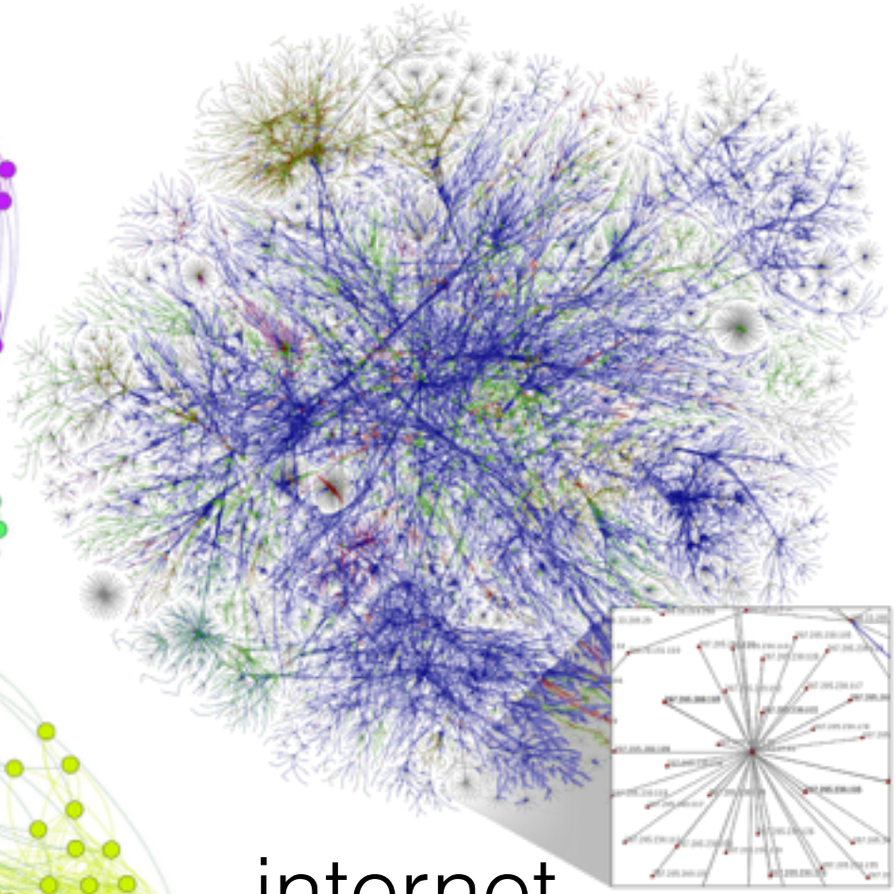
Dense vs. Sparse Networks

A network for which the density ρ tends to a constant as n —> ∞ is **dense.**

A network in which ρ —> 0 as n —> ∞ is **sparse** (the case for most networks).



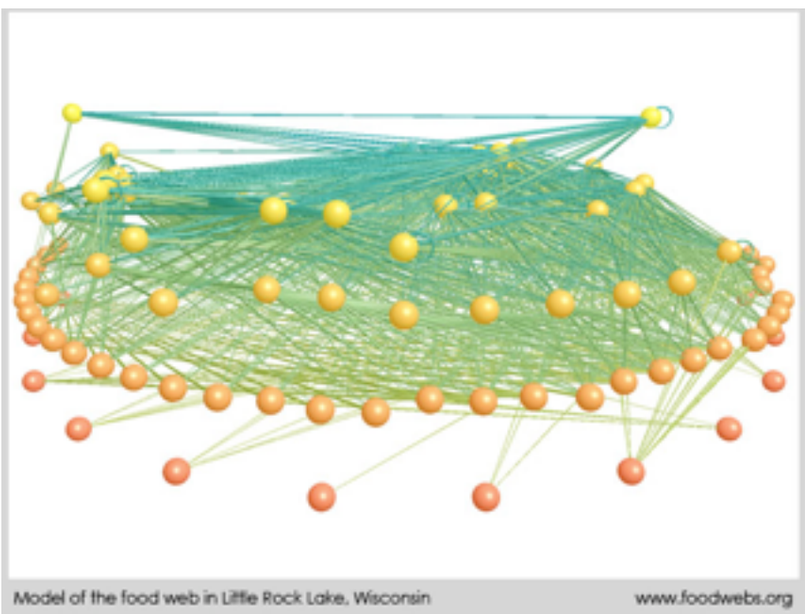
friendship



internet

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

Food webs: density tends to be constant regardless of size



Average Path Length

The **average path length** is the average shortest path between all pairs of vertices.

The **diameter** of a network is the length of the longest shortest path between two vertices in the network.

$\mathbf{d}_{ij}$ denotes the **geodesic distance** from vertex i to vertex j.

The **mean geodesic** is:

$$L = \frac{1}{\frac{n(n+1)}{2}} \sum_{i \geq j} d_{ij}$$

When analyzing disconnected networks, the **harmonic mean** of the geodesics (**global efficiency**) is:

$$L^{-1} = \frac{1}{\frac{n(n+1)}{2}} \sum_{i \geq j} \frac{1}{d_{ij}}$$

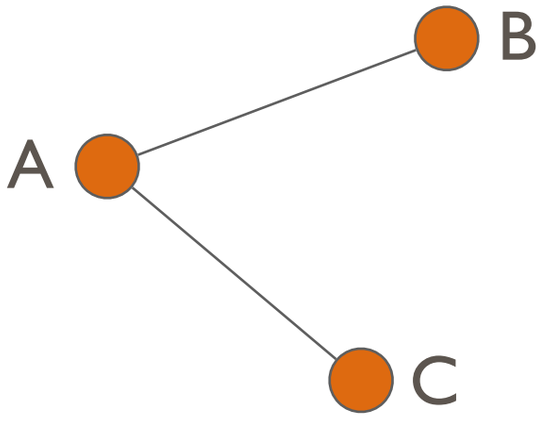
Network Clustering

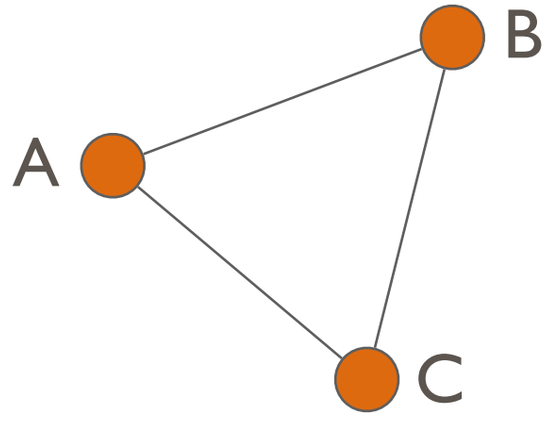
Network clustering (or **transitivity**) is the probability that two neighbors of a vertex will also connect to each other.

Networks with high transitivity are considered to have **local structure**.

A **connected triple** is a set of three nodes A, B, and C, such that A is connected to both B and C.

A **triangle** is a set of three nodes A, B, and C, such that all three are connected to each other.





Clustering coefficient

The **clustering coefficient** of a network is the fraction of triples that have their third edge filled to form a triangle:

$$C = \frac{3 \times \text{the number of triangles in the network}}{\text{number of connected triples of vertices}}$$

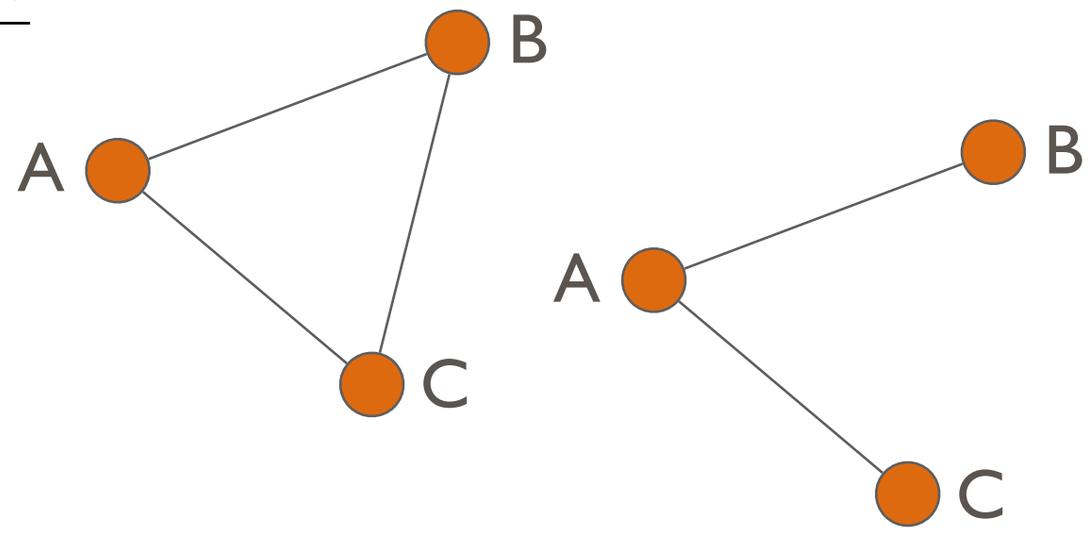
An alternative clustering coefficient starts by calculating the clustering at each node:

$$C_i = \frac{\text{number of triangles connected to vertex } i}{\text{number of triples centered on vertex } i}$$

$C_i = 0$ for nodes with degree 0 or 1

$$C_{WS} = \frac{1}{n}\sum_i C_i$$

weights low degree vertices more heavily



Network Models

Empirical Data

Recent work on social networks within mathematics and physics has focused on three distinctive features of network structure

1. The “small world” effect (a combination of short paths and social structure)
2. The probability that two of your friends know one another is much greater than the probability that two people chosen randomly from the population know each other (clustering)
3. A skewed degree distribution

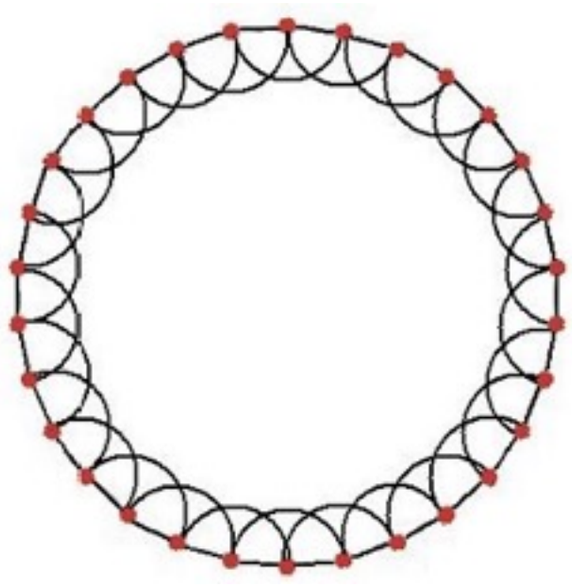
Random graph models of social networks can provide a baseline against which real-world networks can be compared

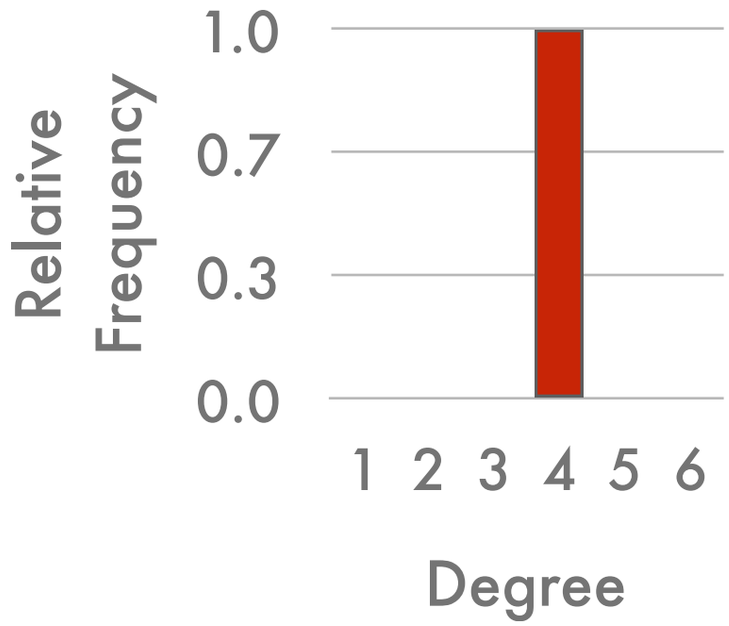
Lattice networks

Homogeneous degree distributions (**regular graphs**)

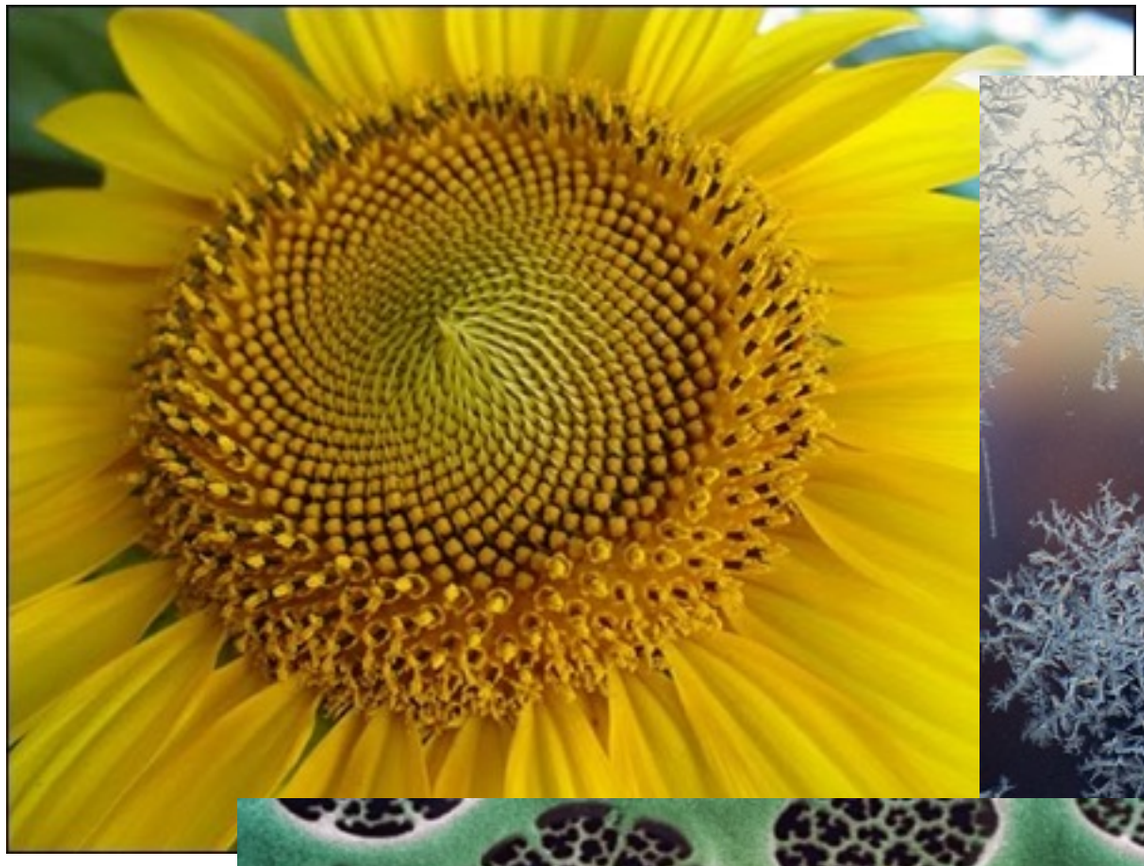
Regular graphs in which all vertices have degree *k* are called **k-regular**

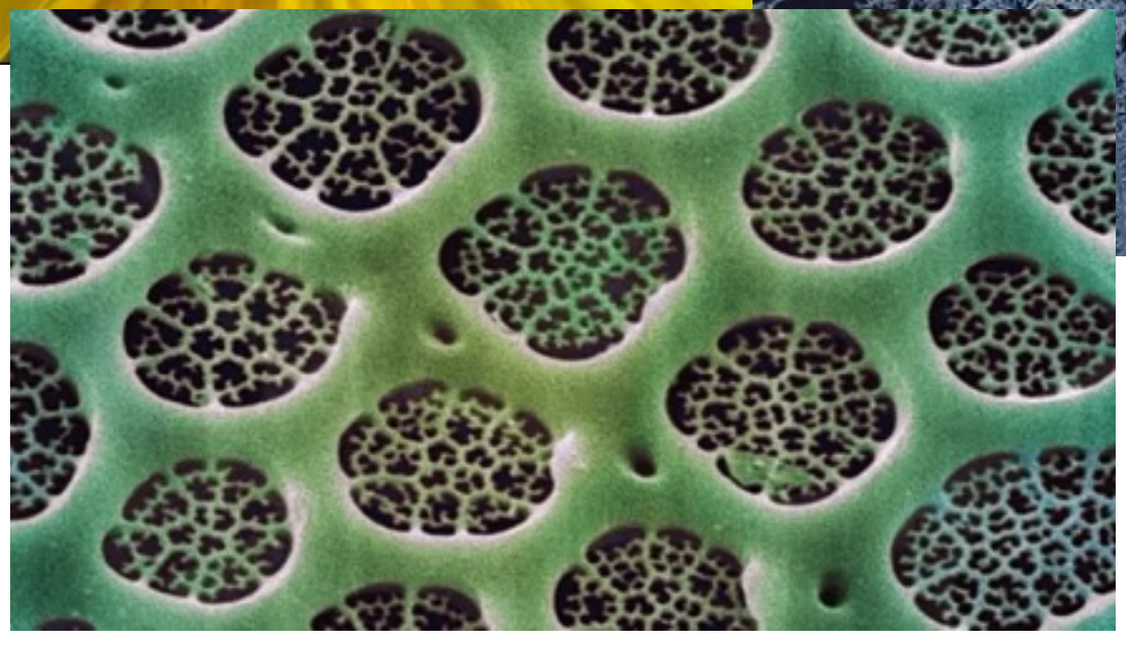
Spatially determined - edges link nearby vertices.





Lattices in nature





Erdös-Rényi random network (1959)

1. Create *n* vertices

2. For each pair of vertices *i* and *j*, create an edge (*i, j*) with probability *p*. The vertices will remain unconnected with probability 1- *p*.

Each node has a degree between 0 and n-1

$$\Pr\{\text{degree } k\} = \Pr\{Y = k\} = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

binomial degree distribution

Erdös-Rényi random network

The structure of the network depends on *p.*

Random connections are non-spatial.

A large random graph has a **Poisson degree distribution.**

Its homogeneous degree distribution makes it a poor approximation of real-world networks but many of its features can be calculated exactly.

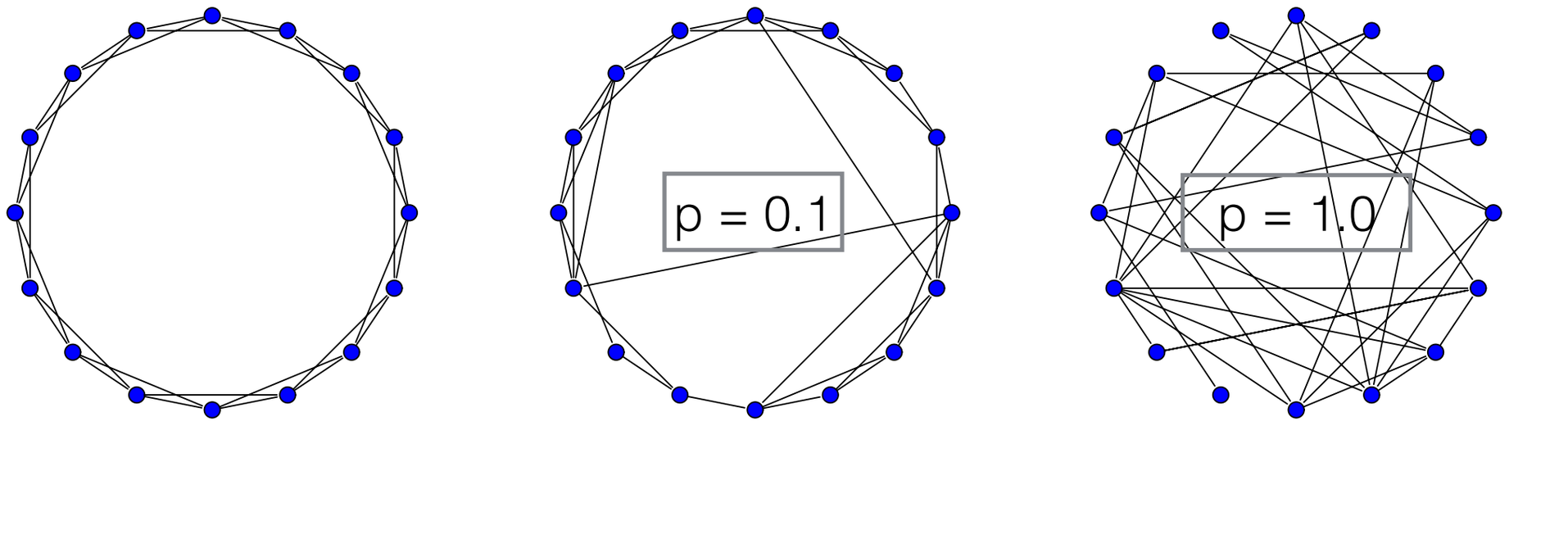


Figure by L.A. Meyers

The Configuration Model

Specify a network size *n*

For each vertex *i:*

1. Choose a degree k_i (can be selected randomly from a specified degree distribution)
2. attach k_i stubs (edges-to-be) to *i*

Choose pairs of stubs at random and connect them together

This produces a graph with exactly the desired degree distribution, but is in all other respects random.

The configuration model is the set of networks produced this way, each having equal weight

The Small World Effect

What is the average distance between two people?

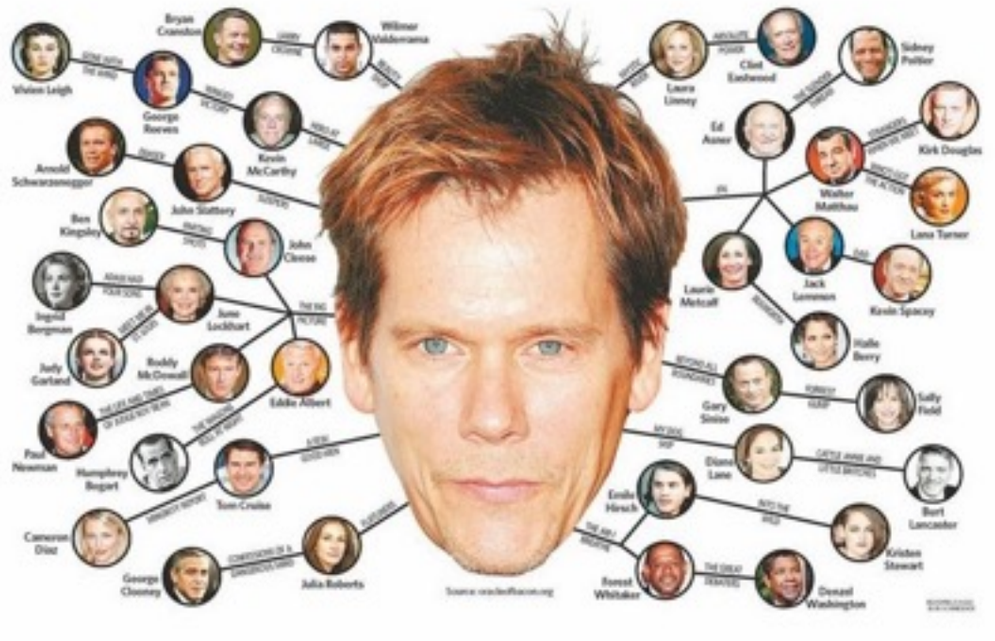
Stanley Milgram's experiment (1967):

- 296 arbitrarily-selected letter "senders" in Boston and Omaha

- Ask "sender" to generate acquaintance chains to target a person in Boston ("the small world method")

- Mean number of intermediaries= 5.2 ("six degrees of separation")

- 48% of chains passed through 3 people



Small world effect: most pairs of vertices in most networks are connected by a short path.

The Small World Model

Watts and Strogatz (1998) developed a simple model for the coexistence of clustering and small average path length.

Start with a one-dimensional ring lattice with *n* nodes where every node is connected to all nodes *k* or fewer steps away.

Rewire the network: For each edge, move one end to a new random location with probability p_r.

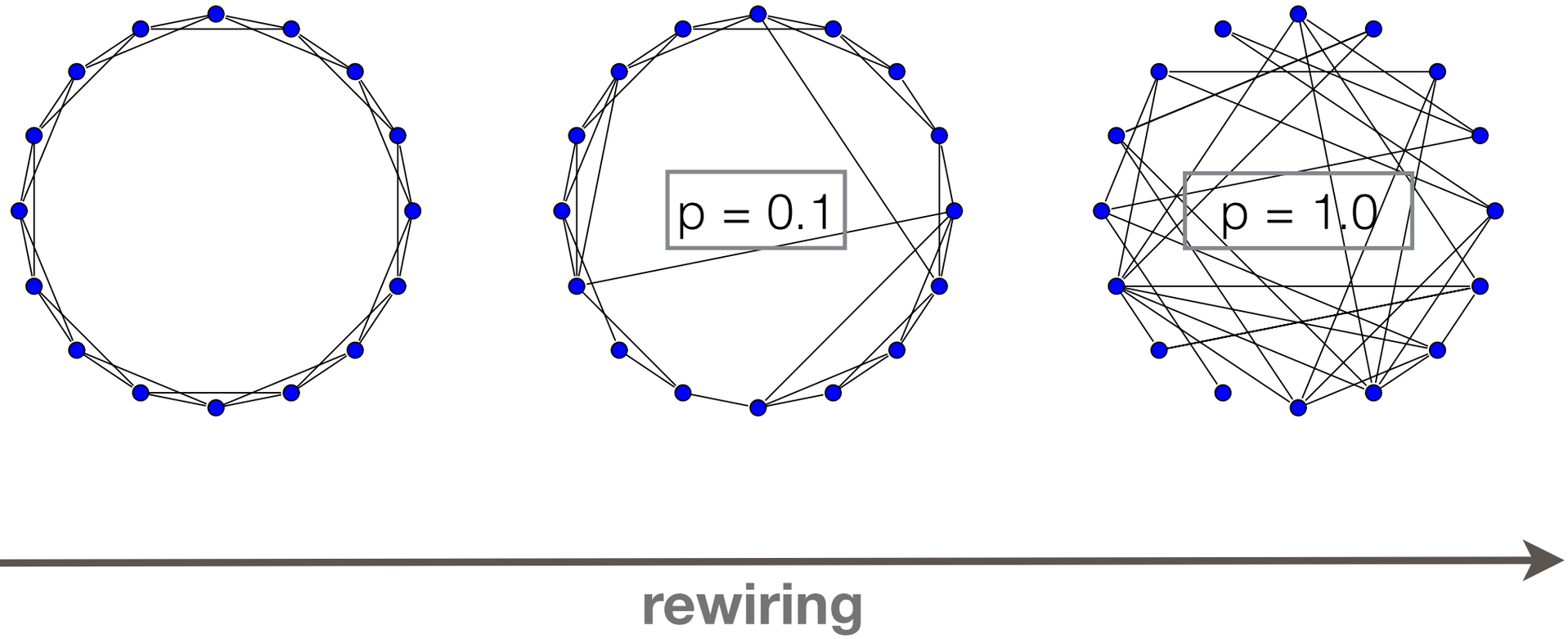


Figure by L.A. Meyers

The Small World Model

Clustering is unaffected by the addition of a few shortcuts

Average path length decreases dramatically with a few shortcuts

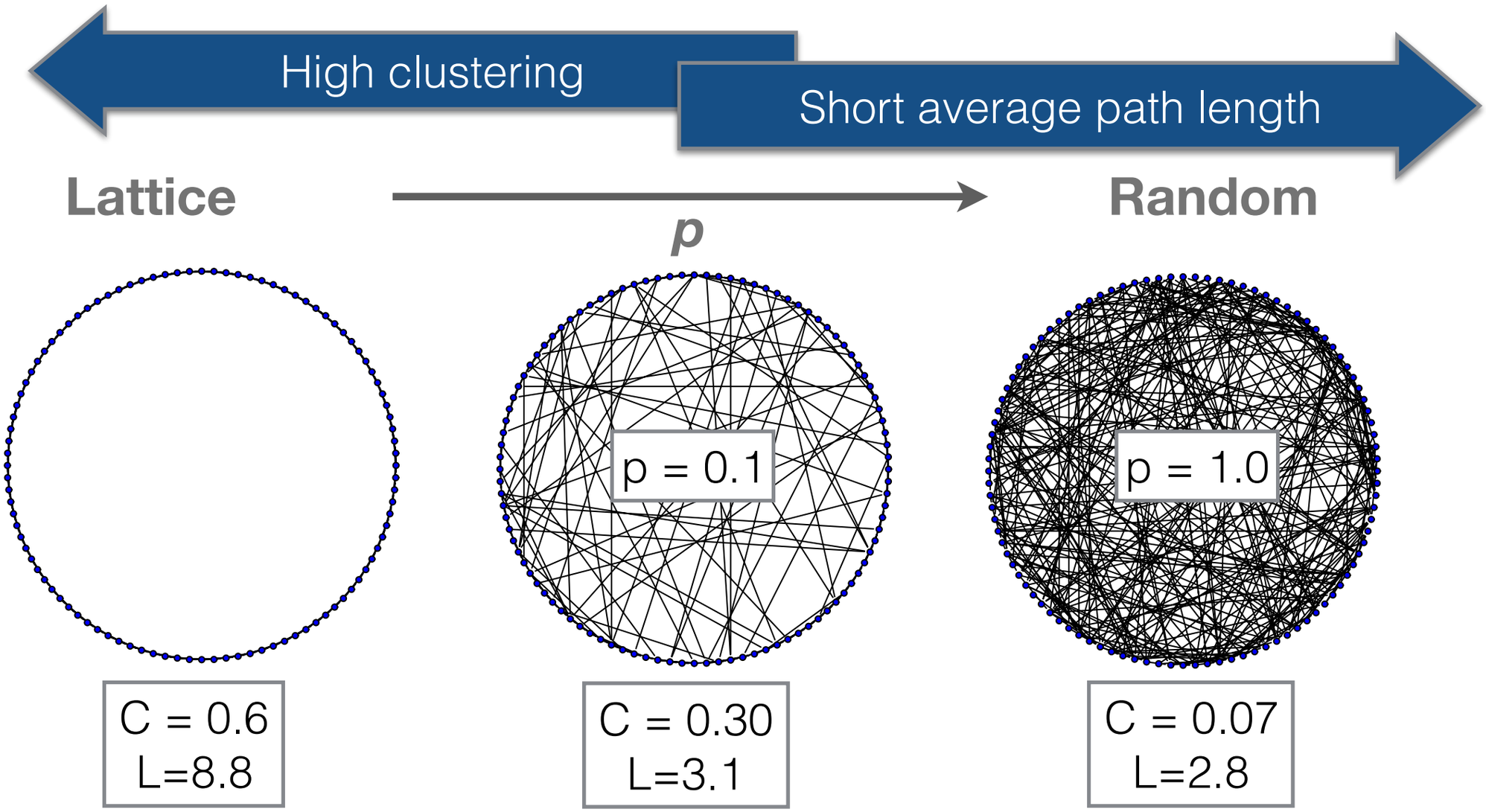
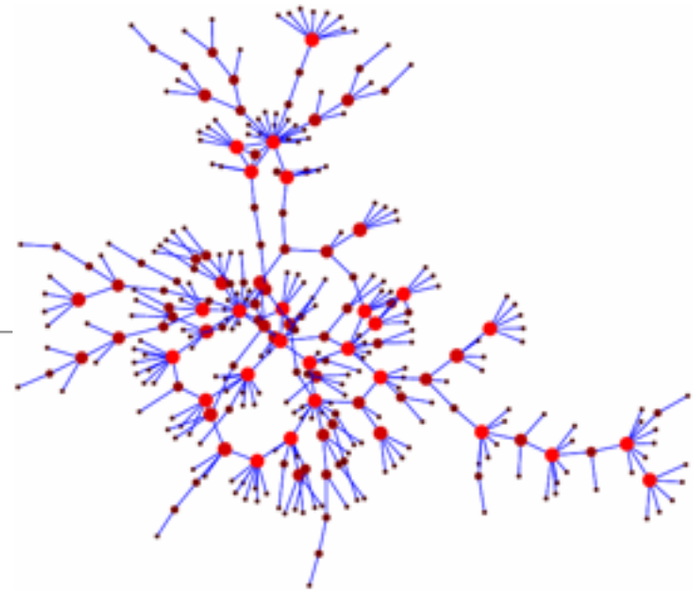


Figure by L.A. Meyers

Scale Free Networks



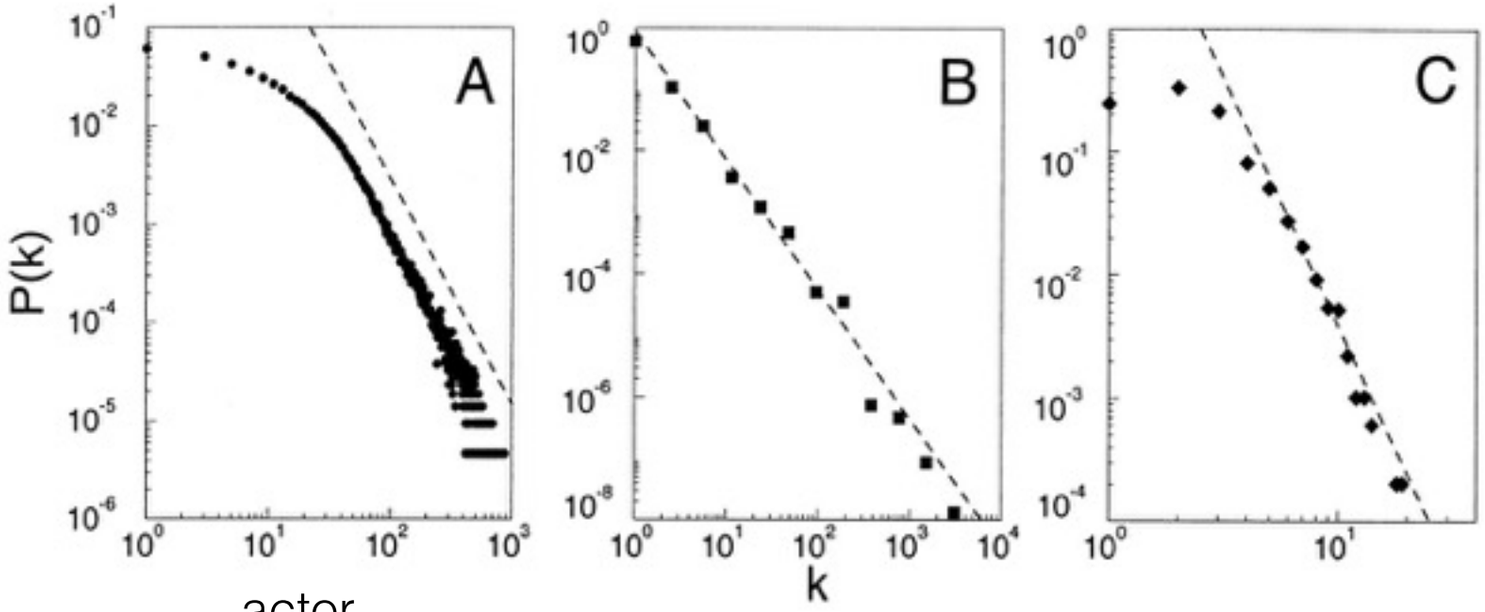
Scale free networks have power law degree distributions. $p_k \sim k^{-\gamma}$

They are also called **power law networks**.

The vast majority of vertices have very low degree (**spokes**) while a small number of vertices have high degree (**hubs**).

Quick test for scale free network: make a log-log plot of the CDF and look for a straight line.

CDF



actor collaboration

www

power grid data

Empirical networks show deviations from strict mathematical degree distributions and are often only power law in the tail of the distribution.

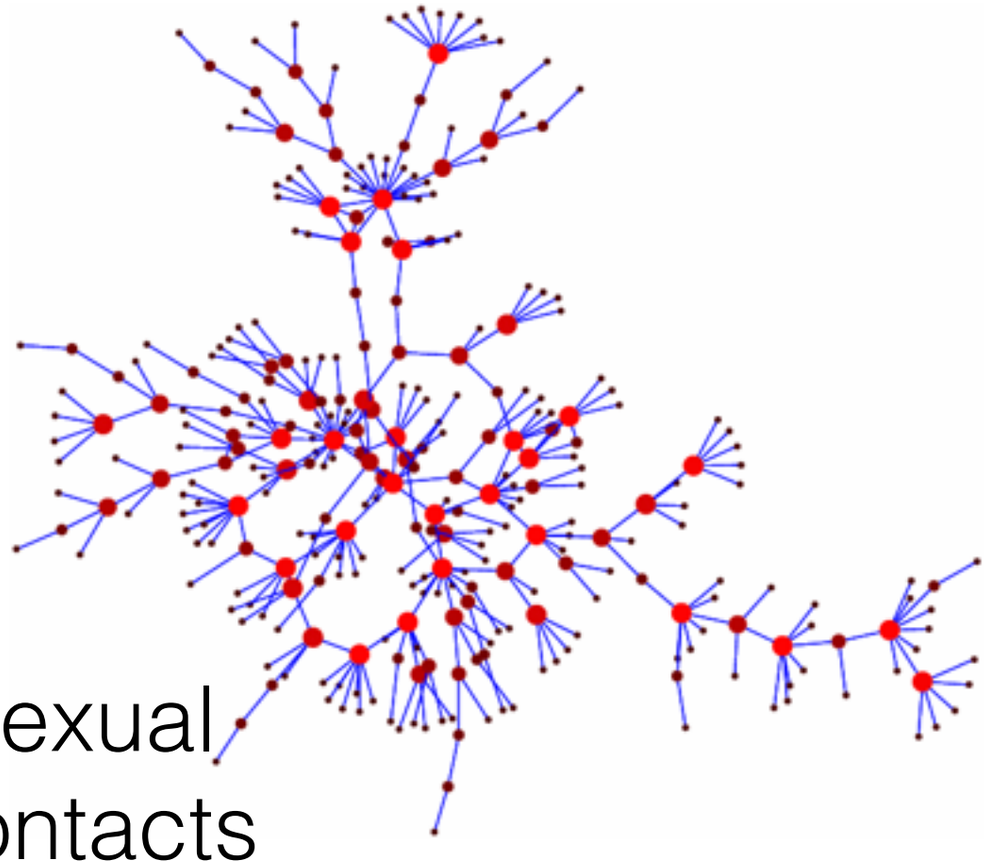
Barabási, Albert-László, and Réka Albert. "Emergence of scaling in random networks." Science 286.5439 (1999): 509-512.

Why are networks scale free?

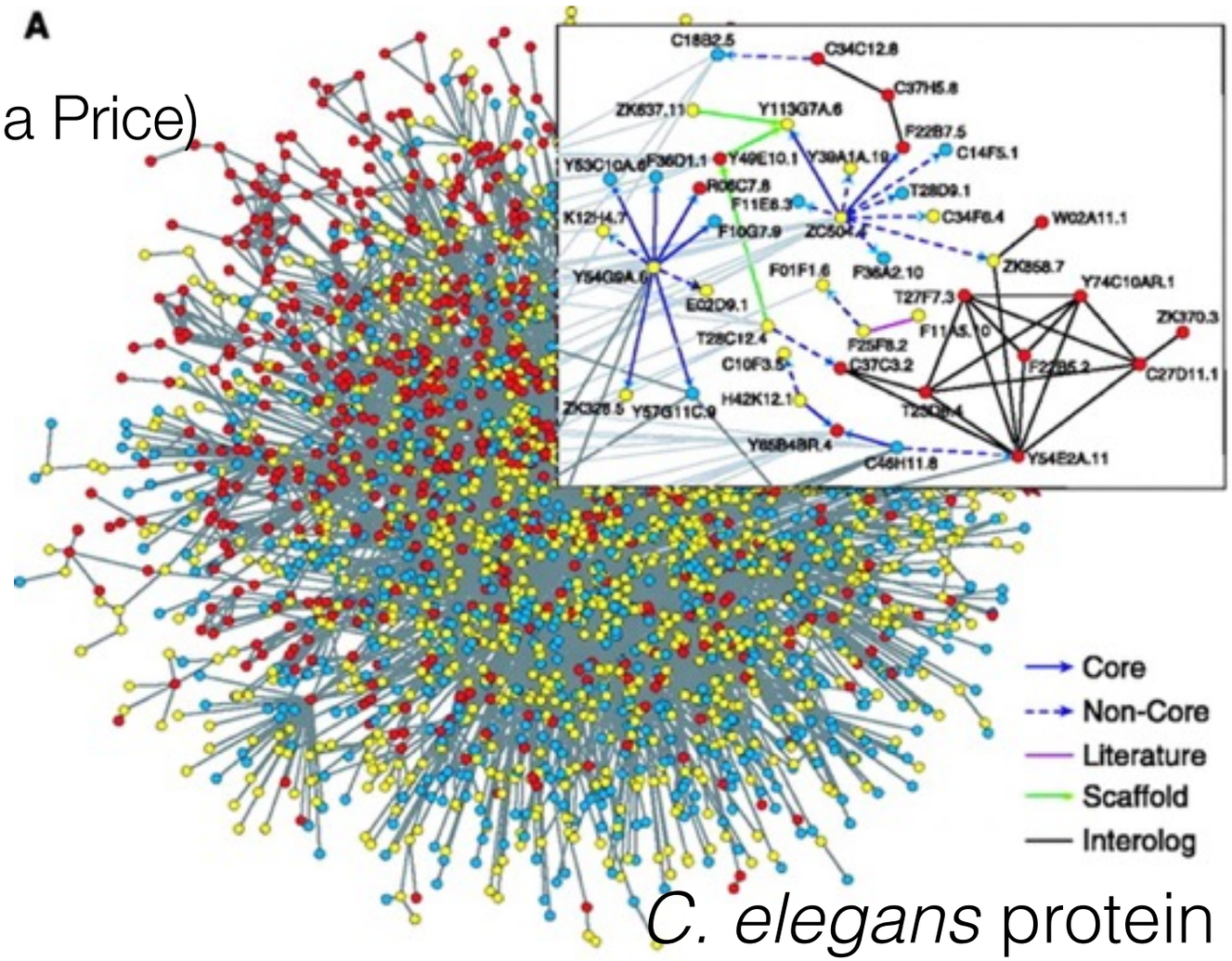
What natural processes potentially give rise to networks with power law degree distributions?

“The rich get richer” (Herbert Simon)

Cumulative advantage (Derek de Solla Price)



sexual contacts



C. elegans protein interaction network

Barabási-Albert Model (1999)

The **Barabási-Albert Model** of preferential attachments describes a simple and realistic process that produces scale free networks.

Growth: the network grows by adding vertices as a function of time.

Preferential attachment: edges are attached to existing vertices chosen at random weighted by the degree of each vertex.

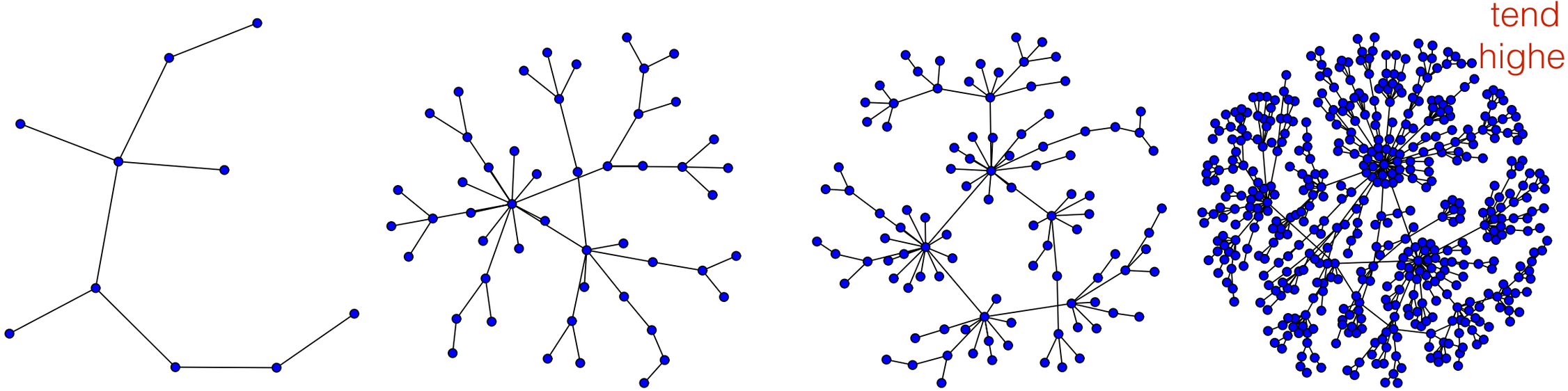


Figure by L.A. Meyers

Newman, Watts, & Strogatz 2002

We can create random networks in which the degree distributions are the same as those for real-world networks, but connections between vertices are otherwise random

If the real-world networks are effectively random, we would expect the predictions of models to agree well with empirical measurements

When agreement isn’t perfect, there is potentially non-trivial structure structure in these networks

Discrepancies between the real-world network data and model predictions indicate nonrandom social phenomena at work in shaping the network