Fundamental Statistics of Graphs: Connectivity

Mark S. Handcock

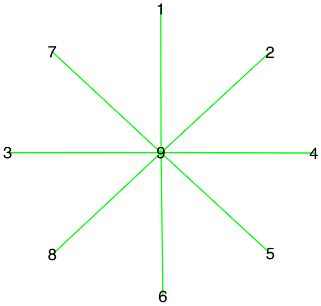
handcock@ucla.edu

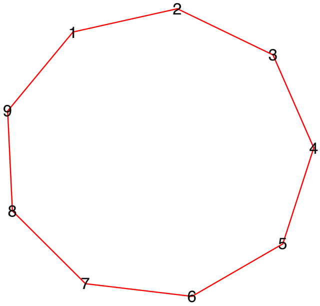
Statistical Analysis of Networks
Based on Notes by Peter D. Hoff
Stat 218

Network connectivity

Density (or average degree) is a very coarse description of a graph.

Compare the n-star graph to the n-circle graph below:





The two graphs have roughly the same density, but the structure is very different.

Evaluating connectivity

Recall, density is the average degree divided by $(n-1)$.

What is the average degree of the

- n-star graph?
- the n circle graph?

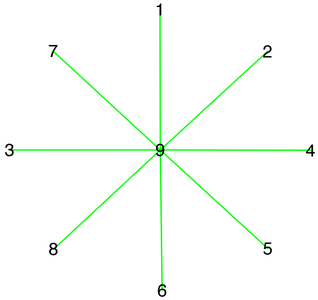
For the circle graph,

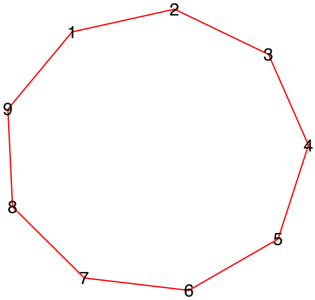
$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i = \frac{1}{n}2n = 2$$

For the star graph,

$$\begin{aligned}
\bar{d} &= \frac{1}{n}\sum_{i=1}^{n} d_i \\
&= \frac{1}{n}((n-1) + 1 + \cdots + 1) \\
&= \frac{1}{n}((n-1) + (n-1)) \\
&= 2\frac{n-1}{n} \approx 2 \quad \text{for large } n
\end{aligned}$$

Evaluating connectivity





Which graph seems more "connected" ?

- The star graph?
  - Each node is within at most two links of every other node.
  - Transmitting information in this network is easier than in the circle graph.
- The circle graph?
  - Removal of one node can completely disconnect the star graph.

Evaluating connectivity

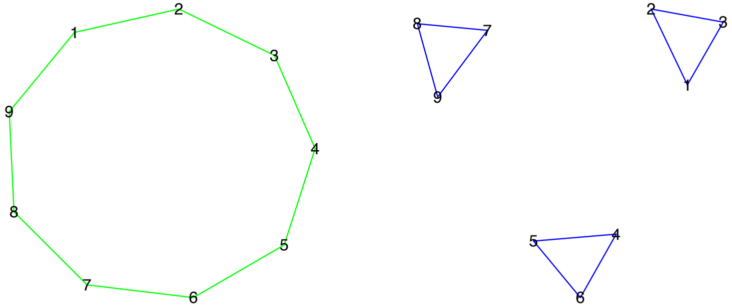
What summary statistics can distinguish between the graphs?
How about degree variability?

Circle graph: $\mathrm{Var}(d_1, \ldots, d_n) = 0$.

Star graph: $\mathrm{Var}(d_1, \ldots, d_n)$ grows linearly with n.

So degree variance can distinguish between these graphs.

Evaluating connectivity



- What is the degree variance for each graph?
- Which one is more "connected"?

Evaluating connectivity

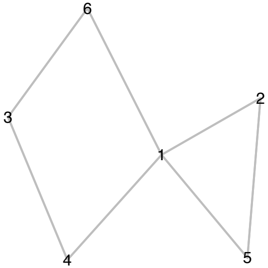
Intuitively, a “highly connected” graph is one in which nodes can reach each other via connections, or a “path.”

To evaluate connectivity (and a variety of other statistics) it will be useful to calculate the length of the shortest path between each pair of nodes.

Walks, trails and paths

Walk: A walk is any sequence of adjacent nodes.

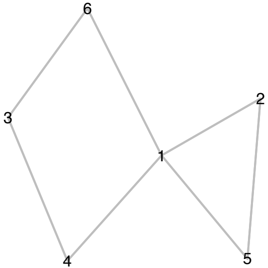
Length of a walk: The number of nodes in the sequence, minus one.



Identify the following walks on the graph:

- $w = (2, 1, 6, 3, 4)$
- $w = (2, 1, 6, 3, 4, 1, 5)$
- $w = (2, 1, 2, 5, 1, 4)$

Trail: A trail is a walk on distinct edges.

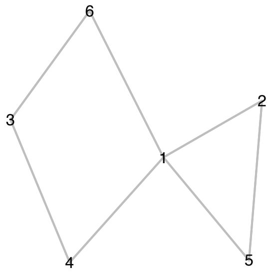


Which of the following are trails on the graph?

- $w = (2, 1, 6, 3, 4)$
- $w = (2, 1, 6, 3, 4, 1, 5)$
- $w = (2, 1, 2, 5, 1, 4)$

Walks, trails and paths

Path: A path is a trail consisting of distinct nodes.



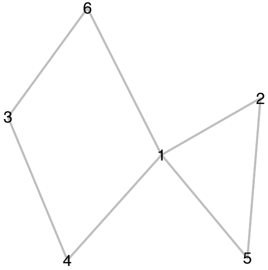
Which of the following are paths on the graph?

- $w = (2, 1, 6, 3, 4)$
- $w = (2, 1, 6, 3, 4, 1, 5)$
- $w = (2, 1, 2, 5, 1, 4)$

Walks, trails and paths

By these definitions,

- each path is a trail,
- each trail is a walk.



Depending on the application, we may be interested in the numbers and kinds of walks, trails and paths between nodes:

- probability: random walks on graphs;
- transport along a network: trails on graphs;
- communication or disease transmission: number of paths between nodes.

To evaluate connectivity, identifying paths will be most useful.

Reachability and connectedness

Reachable: Two nodes are **reachable** if there is a path between them.

Connected: A network is **connected** if every pair of nodes is reachable.

Component: A network **component** is a maximal connected subgraph.

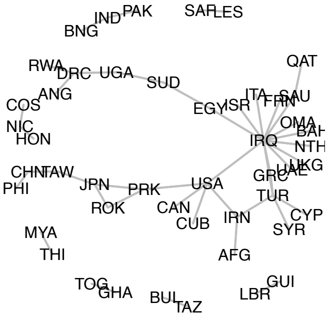
A "maximal connected subgraph" is a connected node-generated subgraph that becomes unconnected by the addition of another node.

An unconnected graph

Symmetrized conflict data: with isolates removed

```
Y<-conflict90s$conflicts
Y<-1*( Y*t(Y)>0 )

deg<-apply(Y,1,sum,na.rm=TRUE)
Y<-Y[ deg>0 ,deg>0 ]
```

Identify all connected components:

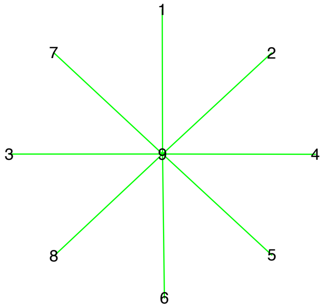


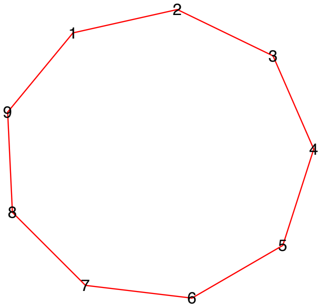
Connectivity and cutpoints

How connected is a graph?

One notion of connectivity is robustness to removal of nodes or edges.

Cutpoint: Let $\mathcal{G}$ be a graph and $\mathcal{G}_{-i}$ be the node-generated subgraph, generated by $\{1, \ldots, n\} \setminus \{i\}$. Then node i is a **cutpoint** if the number of components of $\mathcal{G}_{-i}$ is larger than the number of components of $\mathcal{G}$.

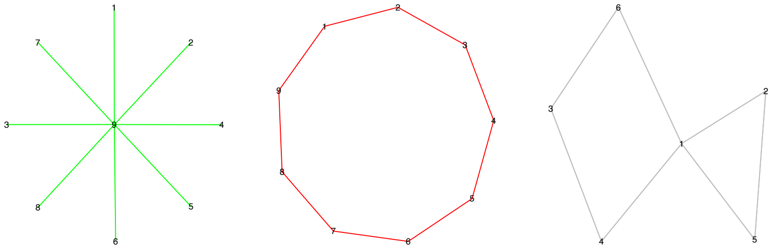




Exercise: Identify any cutpoints of the two graphs.

Node connectivity

Node connectivity: The node connectivity of a graph $k(\mathcal{G})$ is the minimum number of nodes that must be removed to disconnect the graph.



Exercise: Compute the node connectivities of the above graphs.

Limitations of the node connectivity measure

This notion of connectivity is of limited use:

- perhaps most useful in terms of designing robust communication networks;
- less useful for describing the types of networks we've seen.

In particular, node connectivity is a very coarse measure

- it disregards the size of the graph;
- it disregards the number of cutpoints;
- it is of limited descriptive value for real social networks.

Exercise: What is the node connectivity of *all* real graphs we've seen so far?

Average connectivity

Node connectivity is based on a "worst case scenario."

A more representative measure might be some sort of average connectivity.

Connected nodes:
Nodes i,j are connected if there is a path between them.

Dyadic connectivity:
$k(i,j) =$ minimum number of removed nodes required to disconnect i, j.

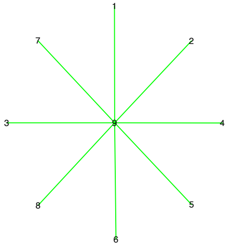
Average connectivity:
$\bar{k} = \sum_{i<j} k(i,j)/\binom{n}{2}$

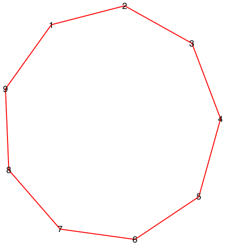
- $\bar{k}$ can be computed in polynomial time;
- bounds on $\bar{k}$ in term of degree, path distances can be obtained.

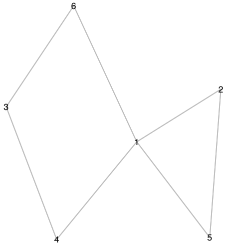
(Beineke, Oellermann, Pippert 2002)

A similar notion of connectivity is to considering robustness to edge removal.

Bridge: Let $\mathcal{G}$ be a graph and $\mathcal{G}_{-e}$ be the graph minus the edge e. Then e is a bridge if the number of components of $\mathcal{G}_{-e}$ is greater than the number of components of $\mathcal{G}$.



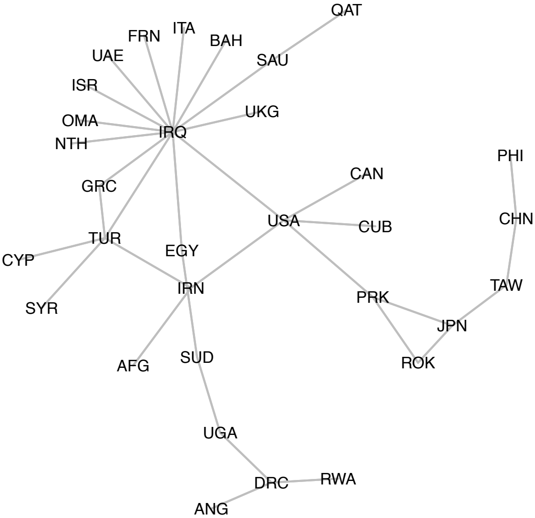




Exercise: Identify some bridges in the above graphs.

Connectivity and bridges

Identify bridges in the big connected component of the conflict network:



Edge connectivity

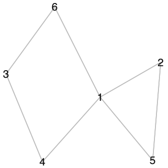
Edge connectivity:
The edge connectivity of a graph is the minimum number of that need to be removed to disconnect the graph.

Like node connectivity, the edge connectivity is of limited use:

- for many real-life graphs, the edge connectivity is one;
- averaged versions of connectivity may be of more use.

A **geodesic** in graph theory is just a shortest path between two nodes.

The **geodesic distance** $d(i, j)$ between nodes i and j is the length of a shortest path between i and j.



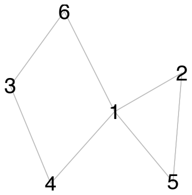
$$\mathbf{D} = \begin{pmatrix} 0 & 1 & 2 & 1 & 1 & 1 \\ 1 & 0 & 3 & 2 & 1 & 2 \\ 2 & 3 & 0 & 1 & 3 & 1 \\ 1 & 2 & 1 & 0 & 2 & 2 \\ 1 & 1 & 3 & 2 & 0 & 2 \\ 1 & 2 & 1 & 2 & 2 & 0 \end{pmatrix}$$

Note: Geodesics are not unique. Consider paths connecting nodes 1 and 3.

Nodal eccentricity

The **eccentricity** of a node is the largest distance from it to any other node:

$$e_i = \max_j d_{i,j}.$$



$$\mathbf{e} = (2, 3, 3, 2, 3, 2)$$

Diameter

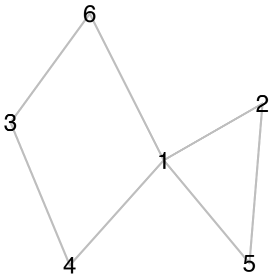
Eccentricities, like degrees, are **node level statistics**.

One common **network level statistic** based on distance is the **diameter**:

The **diameter** of a graph is the largest between-node distance:

$$
\begin{aligned}
\text{diam}(\mathbf{Y}) &= \max_{i,j} d_{i,j} \\
&= \max_i \max_j d_{i,j} \\
&= \max_i e_i
\end{aligned}
$$

Diameter



For our simple six-node example graph,

$$\mathrm{diam}(\mathbf{Y}) = \max\{2, 3, 3, 2, 3, 2\} = 3$$

Diameter and average eccentricity

- For a connected graph, the diameter can range from 1 to $n-1$.
- For an unconnected graph
  - by convention the diameter is taken to be infinity;
  - diameters of connected subgraphs can be computed.
- Like node connectivity, diameter is reflects a "worst case scenario."
  - Average eccentricity, $\bar{e} = \sum e_i/n$ may be a more representative measure.
  - When comparing graphs with different numbers of nodes, it is useful to scale by $n-1$.

A recommendation:

$$\frac{1}{n-1}\bar{e} = \frac{1}{n(n-1)}\sum e_i$$

Counting walks between nodes

Enumerating the number and types of walks between nodes is useful:

- existence of walks between nodes tells us about connectivity.
- existence of walks of minimal length tells us about geodesics.

Walks of all lengths between nodes can be counted using matrix multiplication.

General matrix multiplication: Let

- $\mathbf{X}$ be an $l \times m$ matrix ;
- $\mathbf{Y}$ be an $m \times n$ matrix.

The matrix product $\mathbf{XY}$ is the $l \times n$ matrix $\mathbf{Z}$, with entries

$$z_{i,j} = \sum_{k=1}^{m} x_{i,k} y_{k,j}$$

Useful note: The entries of $\mathbf{Z}$ are dot products of rows of $\mathbf{X}$ with columns of $\mathbf{Y}$.

$$\mathbf{XY} = \begin{pmatrix} \mathbf{x}_1 & \rightarrow \\ \mathbf{x}_2 & \rightarrow \\ \mathbf{x}_3 & \rightarrow \end{pmatrix} \begin{pmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \mathbf{y}_4 \\ \downarrow & \downarrow & \downarrow & \downarrow \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 \cdot \mathbf{y}_1 & \mathbf{x}_1 \cdot \mathbf{y}_2 & \mathbf{x}_1 \cdot \mathbf{y}_3 & \mathbf{x}_1 \cdot \mathbf{y}_4 \\ \mathbf{x}_2 \cdot \mathbf{y}_1 & \mathbf{x}_2 \cdot \mathbf{y}_2 & \mathbf{x}_2 \cdot \mathbf{y}_3 & \mathbf{x}_2 \cdot \mathbf{y}_4 \\ \mathbf{x}_3 \cdot \mathbf{y}_1 & \mathbf{x}_3 \cdot \mathbf{y}_2 & \mathbf{x}_3 \cdot \mathbf{y}_3 & \mathbf{x}_3 \cdot \mathbf{y}_4 \end{pmatrix}$$

Computing comemberships with matrix multiplication

Let $\mathbf{Y}$ be an $n \times m$ affiliation network:

$$y_{i,j} = \text{ membership of person } i \text{ in group } j$$

$$\mathbf{Y} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Transpose: The transpose of an $n \times m$ matrix $\mathbf{Y}$ is the $m \times n$ matrix $\mathbf{X} = \mathbf{Y}^T$ with entries $x_{i,j} = y_{j,i}$.

$$\mathbf{Y}^T = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Computing comemberships with matrix multiplication

Exercise: Complete the multiplication of $\mathbf{Y}$ by $\mathbf{Y}^T$:

$$\mathbf{Y}\mathbf{Y}^T = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ & 2 & & & & \\ & & 2 & & & \\ & & & 0 & & \\ & & & & 2 & \\ & & & & & 1 \end{pmatrix}$$

Letting $\mathbf{X} = \mathbf{Y}\mathbf{Y}^T$, we see $x_{i,j}$ is the number of comemberships of nodes i and j.

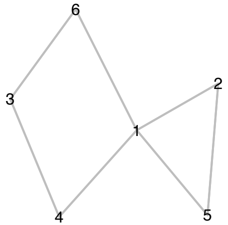
Computing comemberships with matrix multiplication

Exercise: Compute $\mathbf{Y}^T\mathbf{Y}$, and identify what it represents.

$$\mathbf{Y}^T\mathbf{Y} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Multiplying binary sociomatrices

Repeated multiplication of a sociomatrix by itself identifies walks.

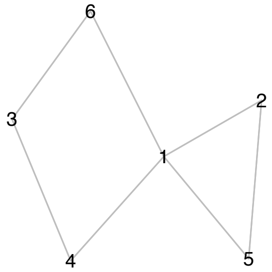



```
Y

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    1    0    1    1    1
## [2,]    1    0    0    0    1    0
## [3,]    0    0    0    1    0    1
## [4,]    1    0    1    0    0    0
## [5,]    1    1    0    0    0    0
## [6,]    1    0    1    0    0    0
```

```
Y %*% Y

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    4    1    2    0    1    0
## [2,]    1    2    0    1    1    1
## [3,]    2    0    2    0    0    0
## [4,]    0    1    0    2    1    2
## [5,]    1    1    0    1    2    1
## [6,]    0    1    0    2    1    2
```

Note: We have replaced the diagonal with zeros for this calculation.

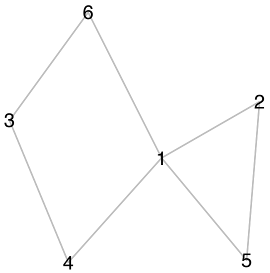
Multiplying binary sociomatrices




```
Y %*% Y

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    4    1    2    0    1    0
## [2,]    1    2    0    1    1    1
## [3,]    2    0    2    0    0    0
## [4,]    0    1    0    2    1    2
## [5,]    1    1    0    1    2    1
## [6,]    0    1    0    2    1    2
```

- How many walks of length 2 are there from i to i?
- How many walks of length 2 are there from i to j?

Multiplying binary sociomatrices




```
Y %*% Y %*% Y

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    2    5    0    6    5    6
## [2,]    5    2    2    1    3    1
## [3,]    0    2    0    4    2    4
## [4,]    6    1    4    0    1    0
## [5,]    5    3    2    1    2    1
## [6,]    6    1    4    0    1    0
```

Result: Let $\mathbf{W} = \mathbf{Y}^k$. Then

$$w_{i,j} = \# \text{ of walks of length } k \text{ between } i \text{ and } j$$

Application: Assessing reachability and Connectedness

Define $\mathbf{X}^{(k)}, k = 1, \ldots n-1$ as follows:

$$
\begin{aligned}
\mathbf{X}^{(1)} &= \mathbf{Y} \\
\mathbf{X}^{(2)} &= \mathbf{Y} + \mathbf{Y}^2 \\
&\vdots \\
\mathbf{X}^{(k)} &= \mathbf{Y} + \mathbf{Y}^2 + \cdots + \mathbf{Y}^k
\end{aligned}
$$

Note:

- $\mathbf{X}^{(1)}$ counts the number of walks of length 1 between nodes;
- $\mathbf{X}^{(2)}$ counts the number of walks of length ≤ 2 between nodes;
- $\mathbf{X}^{(k)}$ counts the number of walks of length $\leq k$ between nodes.

Application: Assessing reachability and Connectedness

Recall:
If two nodes are reachable, there must be a path (walk) between them of length less than or equal to $n - 1$.

Result:
Nodes i and j are reachable if $\mathbf{X}^{(n-1)}_{[i,j]} > 0$.

Recall:
A graph is connected if all pairs are reachable.

Result:
A graph is connected if $\mathbf{X}^{(n-1)}_{[i,j]} > 0$ for all i, j.

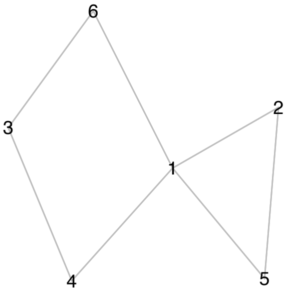
Finding geodesics

Each path is a walk, so

$$\begin{aligned} d_{i,j} &= \text{length of the shortest path between } i \text{ and } j \\ &= \text{length of the shortest walk between } i \text{ and } j \\ &= \text{first } k \text{ for which } \mathbf{Y}^k_{[i,j]} > 0 \end{aligned}$$

This suggests an algorithm for finding geodesic distances.

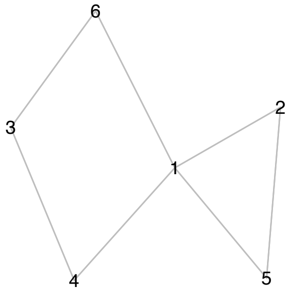
Finding geodesic distances




```
Y

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    1    0    1    1    1
## [2,]    1    0    0    0    1    0
## [3,]    0    0    0    1    0    1
## [4,]    1    0    1    0    0    0
## [5,]    1    1    0    0    0    0
## [6,]    1    0    1    0    0    0
```

$$\mathbf{D} = \begin{pmatrix} 0 & 1 & ? & 1 & 1 & 1 \\ 1 & 0 & ? & ? & 1 & ? \\ ? & ? & 0 & 1 & ? & 1 \\ 1 & ? & 1 & 0 & ? & ? \\ 1 & 1 & ? & ? & 0 & ? \\ 1 & ? & 1 & ? & ? & 0 \end{pmatrix}$$

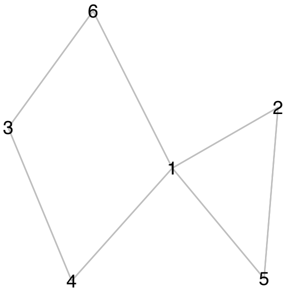
Finding geodesic distances




```
Y %*% Y

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    4    1    2    0    1    0
## [2,]    1    2    0    1    1    1
## [3,]    2    0    2    0    0    0
## [4,]    0    1    0    2    1    2
## [5,]    1    1    0    1    2    1
## [6,]    0    1    0    2    1    2
```

$$\mathbf{D} = \begin{pmatrix} 0 & 1 & 2 & 1 & 1 & 1 \\ 1 & 0 & ? & 2 & 1 & 2 \\ 2 & ? & 0 & 1 & ? & 1 \\ 1 & 2 & 1 & 0 & 2 & 2 \\ 1 & 1 & ? & 2 & 0 & 2 \\ 1 & ? & 1 & 2 & 2 & 0 \end{pmatrix}$$

Finding geodesic distances




```
Y %*% Y %*% Y

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    2    5    0    6    5    6
## [2,]    5    2    2    1    3    1
## [3,]    0    2    0    4    2    4
## [4,]    6    1    4    0    1    0
## [5,]    5    3    2    1    2    1
## [6,]    6    1    4    0    1    0
```

$$\mathbf{D} = \begin{pmatrix} 0 & 1 & 2 & 1 & 1 & 1 \\ 1 & 0 & 3 & 2 & 1 & 2 \\ 2 & 3 & 0 & 1 & 3 & 1 \\ 1 & 2 & 1 & 0 & 2 & 2 \\ 1 & 1 & 3 & 2 & 0 & 2 \\ 1 & 3 & 1 & 2 & 2 & 0 \end{pmatrix}$$

R-function netdist

```
netdist

## function (Y, countdown = FALSE)
## {
##     Y <- 1 * (Y > 0)
##     n <- dim(Y)[1]
##     Y0 <- Y
##     diag(Y0) <- 0
##     Ys <- Y0
##     D <- Y
##     D[Y == 0] <- n + 1
##     diag(D) <- 0
##     s <- 2
##     while (any(D == n + 1) & s < n) {
##         Ys <- 1 * (Ys %*% Y0 > 0)
##         D[Ys == 1] <- ((s + D[Ys == 1]) - abs(s - D[Ys == 1]))/2
##         s <- s + 1
##         if (countdown) {
##             cat(n - s, "\n")
##         }
##     }
##     D[D == n + 1] <- Inf
##     D
## }
## <environment: namespace:rda>
```

R-function `netdist`

```
netdist(Y)

##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    1    2    1    1    1
## [2,]    1    0    3    2    1    2
## [3,]    2    3    0    1    3    1
## [4,]    1    2    1    0    2    2
## [5,]    1    1    3    2    0    2
## [6,]    1    2    1    2    2    0
```

Application: Multidimensional scaling

Often we have data on **distances** or **dissimilarities** between a set of objects.

- Machine learning: $d_{i,j} = |\mathbf{x}_i - \mathbf{x}_j|$, $x_i =$ vector of characteristics of object i.
- Social networks: $d_{i,j} =$ geodesic distance between i and j.

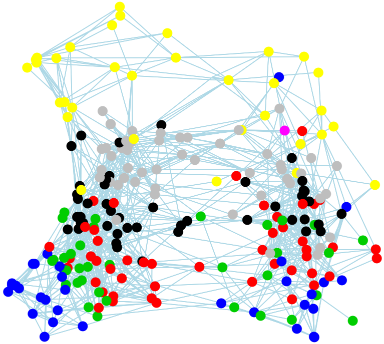
It is often useful to embed these distances in a low-dimensional space.

- visualization: convert distances to a map in 2 dimensions for plotting.
- data reduction: convert $n \times n$ dissimilarity matrix to an $n \times p$ matrix of positions.

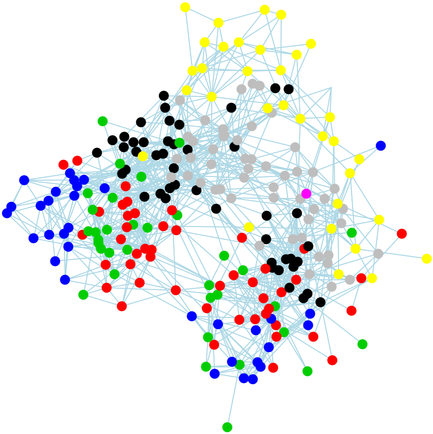
Application: Multidimensional scaling

```
Y<-el2sm(addhealth9$E)
Y<-1*( Y>0 | t(Y)>0 )
D<-netdist(Y)

iso<-which(apply(D==Inf,2,sum) == nrow(Y)-1 )
Y<-Y[-iso,-iso]
D<-D[-iso,-iso]

X<-cmdscale(D)

head(X)

##         [,1]        [,2]
## 1 -0.5547701 -0.27287140
## 2 -0.8151498 -0.58111897
## 3  1.7920205 -0.22866851
## 4 -0.9555775  0.09652415
## 5  0.4975752  0.44596394
## 6  1.2961508 -0.67148355
```

Application: Multidimensional scaling



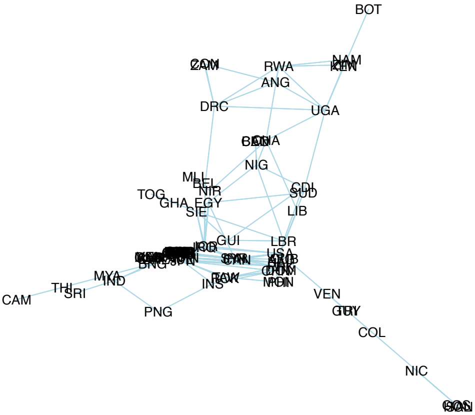
Compare to Fruchterman-Reingold:



Application: MDS for conflict data

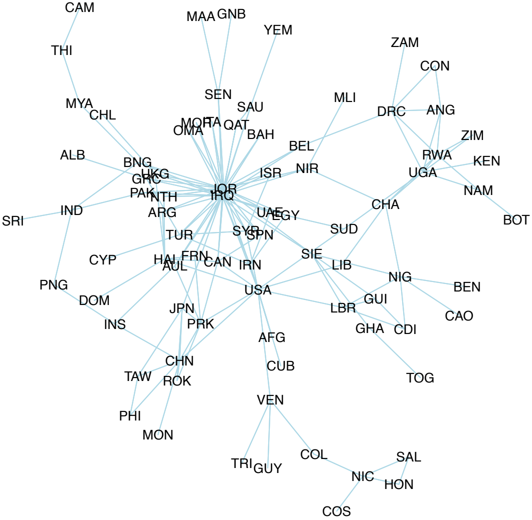
```
Y<-conflict90s$conflicts
Y<-1*( Y>0 | t(Y)>0 )
bigcc<-concomp(Y)[[1]]

Y<-Y[ bigcc ,bigcc ]
D<-netdist(Y)

X<-cmdscale(D)

head(X)

##            [,1]       [,2]
## AFG  0.8961192 -0.4593871
## ALB -1.4510858 -0.4221417
## ANG  0.7930768  2.7900261
## ARG -0.8675232 -0.3523978
## AUL -0.9023903 -0.4603945
## BAH -0.9136724 -0.3166990
```

Application: MDS for conflict data



Application: MDS for conflict data

Compare to Fruchterman-Reingold:



Application: Finding connected components

The distance matrix, or $\mathbf{X}^{(n-1)}$, identifies connected components of a graph.


```
Y

##    1 2 3 4 5 6 7 8 9 10 11 12
## 1  0 1 1 0 0 0 0 0 0  0  0  0
## 2  1 0 1 0 0 0 0 0 0  0  0  0
## 3  1 1 0 1 0 0 0 0 0  0  0  0
## 4  0 0 1 0 1 1 0 0 0  0  0  0
## 5  0 0 0 1 0 1 0 0 0  0  0  0
## 6  0 0 0 1 1 0 0 0 0  0  0  0
## 7  0 0 0 0 0 0 0 1 1  0  0  0
## 8  0 0 0 0 0 0 1 0 1  0  0  0
## 9  0 0 0 0 0 0 1 1 0  1  0  0
## 10 0 0 0 0 0 0 0 0 1  0  1  1
## 11 0 0 0 0 0 0 0 0 0  1  0  1
## 12 0 0 0 0 0 0 0 0 0  1  1  0
```

Application: Finding connected components

The distance matrix, or $\mathbf{X}^{(n-1)}$, identifies connected components of a graph.


```
Y + Y%*%Y

##    1 2 3 4 5 6 7 8 9 10 11 12
## 1  2 2 2 1 0 0 0 0 0  0  0  0
## 2  2 2 2 1 0 0 0 0 0  0  0  0
## 3  2 2 3 1 1 1 0 0 0  0  0  0
## 4  1 1 1 3 2 2 0 0 0  0  0  0
## 5  0 0 1 2 2 2 0 0 0  0  0  0
## 6  0 0 1 2 2 2 0 0 0  0  0  0
## 7  0 0 0 0 0 0 2 2 2  1  0  0
## 8  0 0 0 0 0 0 2 2 2  1  0  0
## 9  0 0 0 0 0 0 2 2 3  1  1  1
## 10 0 0 0 0 0 0 1 1 1  3  2  2
## 11 0 0 0 0 0 0 0 0 1  2  2  2
## 12 0 0 0 0 0 0 0 0 1  2  2  2
```

Application: Finding connected components

The distance matrix, or $\mathbf{X}^{(n-1)}$, identifies connected components of a graph.


```
Y + Y%*%Y + Y%*%Y%*%Y

##    1 2 3 4 5 6 7 8 9 10 11 12
## 1  4 5 6 2 1 1 0 0 0  0  0  0
## 2  5 4 6 2 1 1 0 0 0  0  0  0
## 3  6 6 5 6 2 2 0 0 0  0  0  0
## 4  2 2 6 5 6 6 0 0 0  0  0  0
## 5  1 1 2 6 4 5 0 0 0  0  0  0
## 6  1 1 2 6 5 4 0 0 0  0  0  0
## 7  0 0 0 0 0 0 4 5 6  2  1  1
## 8  0 0 0 0 0 0 5 4 6  2  1  1
## 9  0 0 0 0 0 0 6 6 5  6  2  2
## 10 0 0 0 0 0 0 2 2 6  5  6  6
## 11 0 0 0 0 0 0 1 1 2  6  4  5
## 12 0 0 0 0 0 0 1 1 2  6  5  4
```

R-function concomp

```
concomp

## function (Y)
## {
##     Y0 <- 1 * (Y > 0)
##     diag(Y0) <- 1
##     Y1 <- Y0
##     for (i in 1:dim(Y0)[1]) {
##         Y1 <- 1 * (Y1 %*% Y0 > 0)
##     }
##     cc <- list()
##     idx <- 1:dim(Y1)[1]
##     while (dim(Y1)[1] > 0) {
##         c1 <- which(Y1[1, ] == 1)
##         cc <- c(cc, list(idx[c1]))
##         Y1 <- Y1[-c1, -c1, drop = FALSE]
##         idx <- idx[-c1]
##     }
##     cc[order(-sapply(cc, length))]
## }
## <environment: namespace:rda>
```

R-function concomp

```
concomp(Y)

## [[1]]
## [1] 1 2 3 4 5 6
##
## [[2]]
## [1]  7  8  9 10 11 12

connodes<-concomp(Y)

Y[ connodes[[1]],connodes[[1]] ]

##   1 2 3 4 5 6
## 1 0 1 1 0 0 0
## 2 1 0 1 0 0 0
## 3 1 1 0 1 0 0
## 4 0 0 1 0 1 1
## 5 0 0 0 1 0 1
## 6 0 0 0 1 1 0
```

All these concepts generalize to directed graphs:

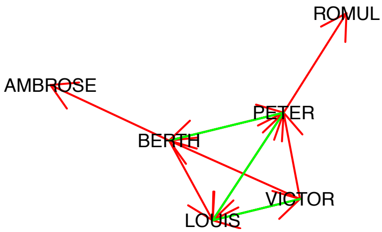
Directed walk: A sequence of nodes $i_1, \ldots, i_K$ such that $y_{i_k, i_{k+1}} = 1$ for $k = 1, \ldots, K-1$.

Powers of the sociomatrix correspond to counts of directed walks.

$$\mathbf{X}^{(k)} = \mathbf{Y} + \mathbf{Y}^2 + \cdots + \mathbf{Y}^k$$

$$x_{i,j}^{(k)} = \text{ \# of directed walks from } i \text{ to } j \text{ of length } k \text{ or less}$$

Example: Praise among monks




```
netdist(Yr)

##         ROMUL AMBROSE BERTH PETER LOUIS VICTOR
## ROMUL       0     Inf   Inf   Inf   Inf    Inf
## AMBROSE   Inf       0   Inf   Inf   Inf    Inf
## BERTH       2       1     0     1     1      2
## PETER       1       2     1     0     1      2
## LOUIS       2       3     2     1     0      1
## VICTOR      2       2     1     1     1      0
```