Social Network Analysis

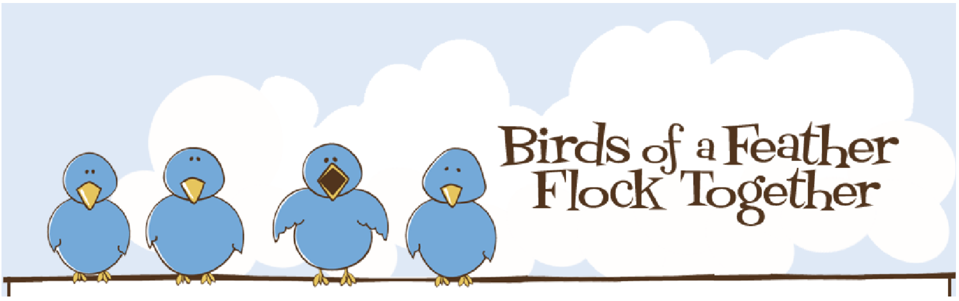
Chen Avin
Communication Systems Engineering

Unit 4
Local Relations:
On Homophily , Assortativity Mixing, Modularity and Structural Balance

(Based on Networks, Crowds, and Markets: Reasoning about a Highly Connected World. By David Easley and Jon Kleinberg. Chapter 4,5 and Networks: An Introduction. By M.E.J Newman)

Homophily

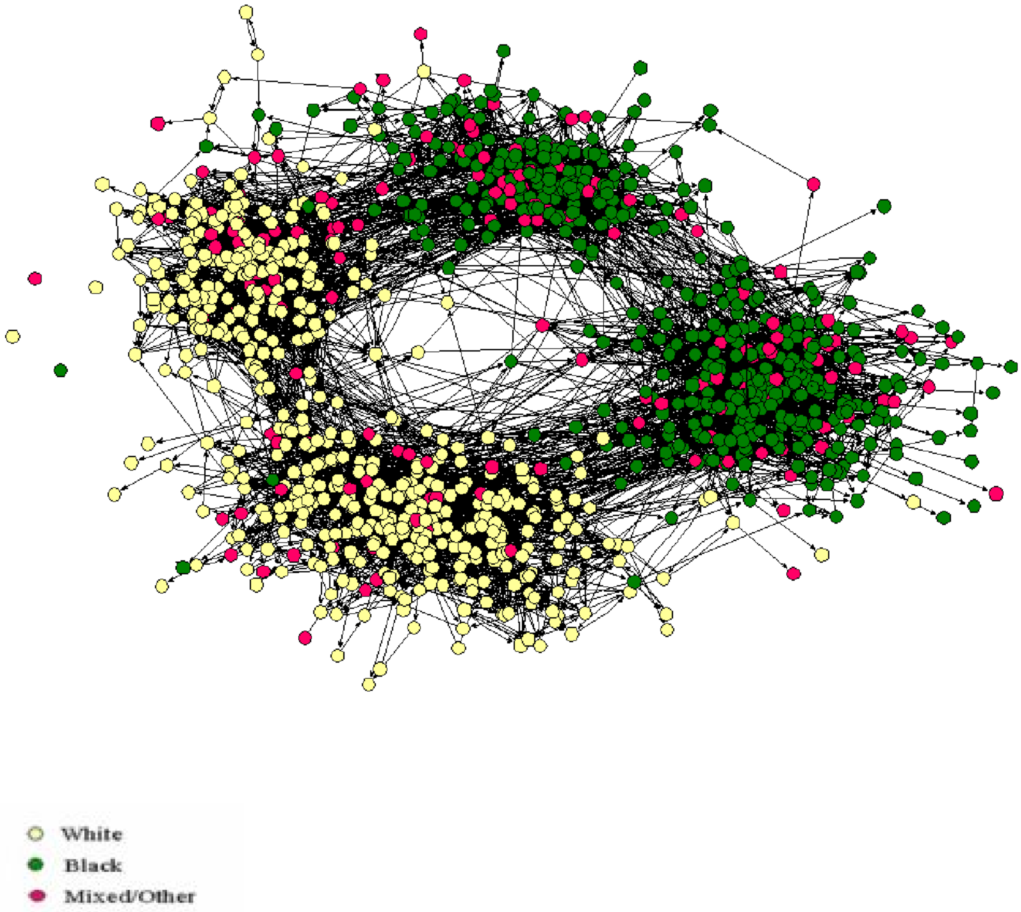
- The principle that we tend to be similar to our friends
  - Plato: “similarity begets friendship”
  - Aristotle: people "love those who are like themselves"
  - "birds of a feather flock together."



  - מָצָא מִין אֶת מִינוֹ
- Link formation can be based both on *intrinsic* (local) or on *context*
  - Intrinsic: A introduce B to C (both C, B trust A and have opportunity to meet)
  - Context: For example All A-B-C work at the same place, or another case: B is similar to A, C is similar to A, so B and C are similar (larger chance to be friends).

Example

High School Students, Homophily by Race (left to right) and Age (top to bottom)

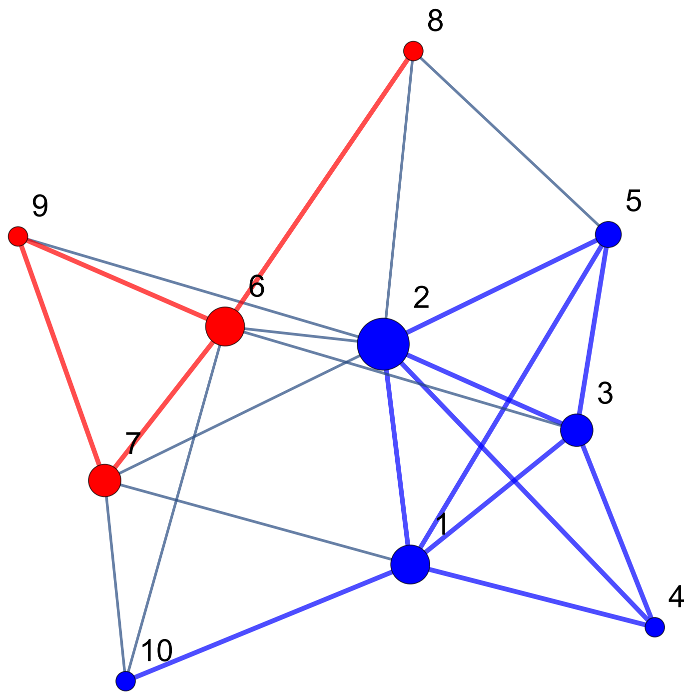


(From : James Moody. Race, school integration, and friendship segregation in America. American Journal of Sociology, 107 (3) : 679–716, November 2001.)

How do we check for Homophily?

Measuring Homophily - Take 1

- For example take a small network and study homophily by gender (color)



- What it means if the network exhibits no homophily?
- Edges will be “random”, independent of nodes color
- Suppose a fraction p are male and a fraction q are female
- We expect p^2 of edges to be male-male q^2 to be female-female and $2pq$ to be mixed
- Here is a natural **Homophily Test**:

If the fraction of cross-gender edges is significantly less than 2pq, then there is evidence for homophily.

- *Inverse homophily* is also possible (hetrophily) - for example in romantic relationships (mostly opposite sex partners and most edges are cross gender)

Let’s check the homophily test

In[31]:=
```
p = N[Length[blue] / VertexCount[EXG]]
```
Out[31]=
```
0.6
```

In[32]:=
```
q = N[Length[red] / VertexCount[EXG]]
```
Out[32]=
```
0.4
```

In[33]:=
```
EdgeCount[EXG]
```
Out[33]=
```
23
```

- Expected Mixed Edges (EME)

In[34]:=
```
2 p q
```
Out[34]=
```
0.48
```

```
EME  = 2 p q EdgeCount[EXG]
```

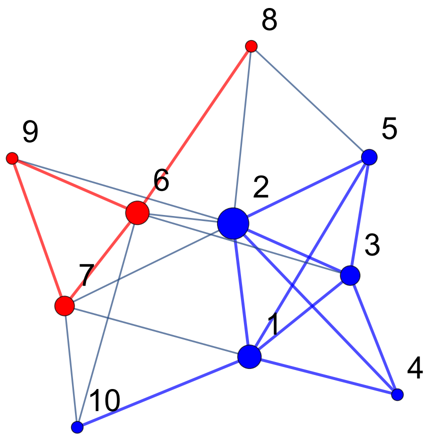
In[35]:=
```
11.040000000000001`
```
Out[35]=
```
11.04
```

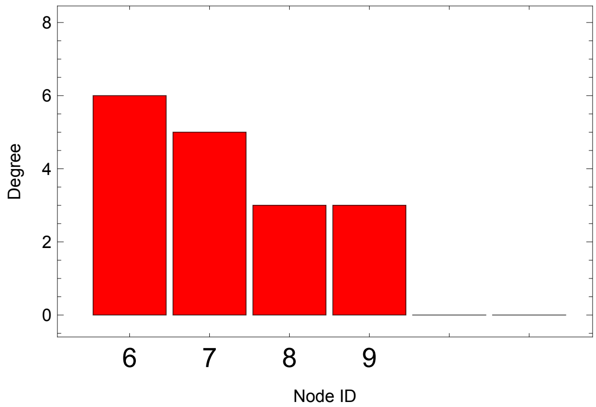
In[36]:=
```
MixedEdge[g_, e_] := If[PropertyValue[{g, e⟦1⟧}, VertexStyle] ===
    PropertyValue[{g, e⟦2⟧}, VertexStyle], 0, 1];
```

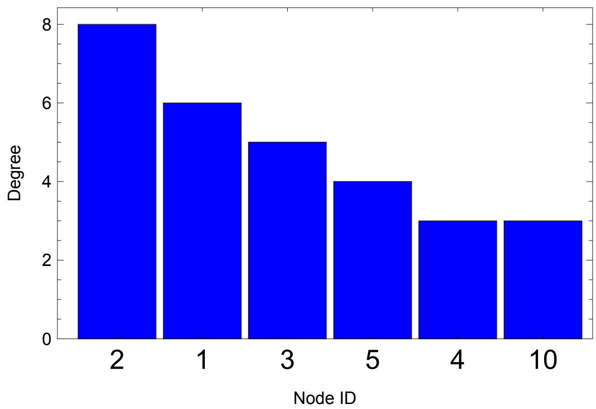
In[37]:=
```
TotalMixedEdges = Total[MixedEdge[EXG, #] & /@ EdgeList[EXG]]
```
Out[37]=
```
9
```

But, it may be the case that Male/Female “control” more edges? One side has most of the edges. Can we fix it?

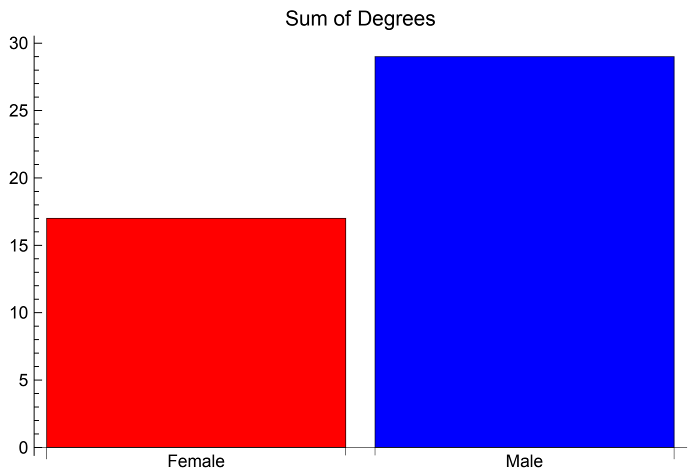
Different degree sequence







Sum of Degrees



Adjust calculation

In[38]:=
```
p1 = N[Total[VertexDegree[EXG][[blue]]] / Total[VertexDegree[EXG]]]
```
Out[38]=
```
0.630435
```

In[39]:=
```
q1 = N[Total[VertexDegree[EXG][[red]]] / Total[VertexDegree[EXG]]]
```
Out[39]=
```
0.369565
```

In[40]:=
```
EdgeCount[EXG]
```
Out[40]=
```
23
```

- Expected Mixed Edges

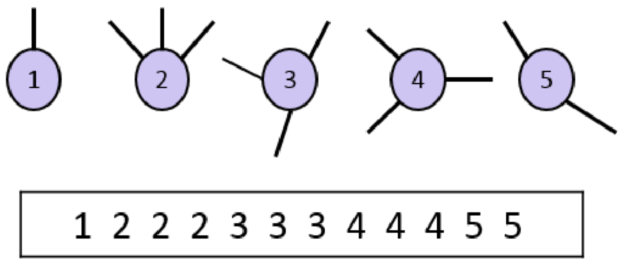
In[41]:=
```
EME = 2 p1 q1 EdgeCount[EXG]
```
Out[41]=
```
10.7174
```

- Recall the actual mixing edges

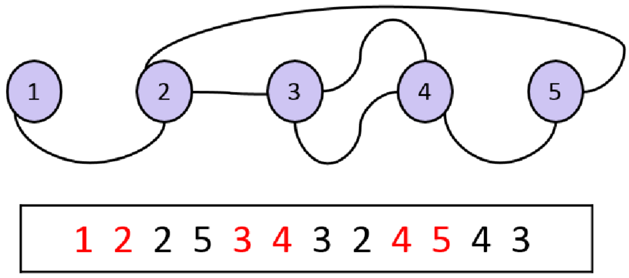
In[42]:=
```
TotalMixedEdges = Total[MixedEdge[EXG, #] & /@ EdgeList[EXG]]
```
Out[42]=
```
9
```

Let' s compare to a random graph with the same degree sequence (configuration model)

Random Configuration Model
1. Set edges



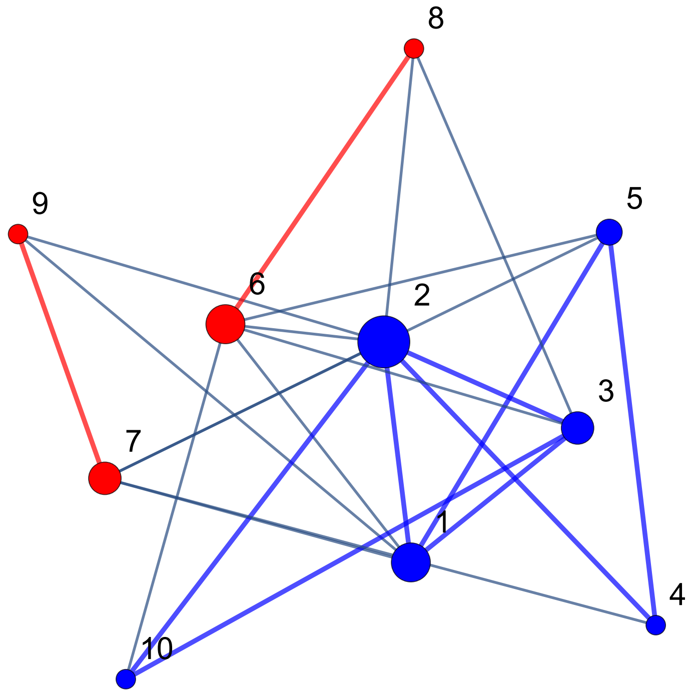
2. Random Matching (here by random order)



In[179]:=

```
cor = PropertyValue[{EXG, #}, VertexCoordinates] & /@ VertexList[EXG];
Deg = VertexDegree[EXG];
blue = {1, 2, 3, 4, 5, 10};
red = {6, 7, 8, 9};
RG = RandomGraph[DegreeGraphDistribution[Deg]];
edgeS = Table[EdgeList[RG][[i]] → If[MemberQ[red, EdgeList[RG][[i]][[1]]] &&
       MemberQ[red, EdgeList[RG][[i]][[2]]], {Red, Thickness[.007]},
      If[MemberQ[blue, EdgeList[RG][[i]][[1]]] && MemberQ[blue, EdgeList[RG][[i]][[2]]],
       {Blue, Thickness[.007]}, Thickness[.004]]], {i, 1, EdgeCount[RG]}];
REXG = Graph[RG, VertexCoordinates → cor,
   VertexStyle → Join[Thread[blue → Blue], Thread[red → Red]], VertexSize →
    Table[VertexList[EG][[i]] → {"Scaled", VertexDegree[EG][[i]] / 6 / EdgeCount[EG]},
     {i, 1, VertexCount[EG]}],
   VertexLabels → "Name", VertexLabelStyle → 16, EdgeStyle → edgeS ]
TotalMixedEdges = Total[MixedEdge[REXG, #] & /@ EdgeList[REXG]]
```

Out[185]=

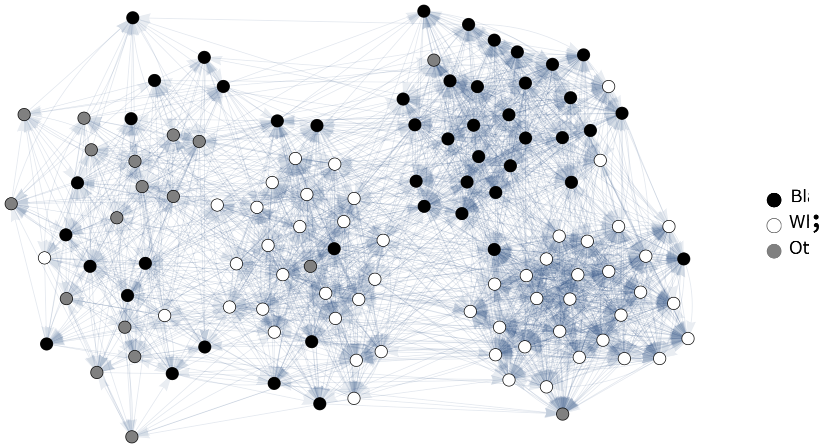


Out[186]=

13

Working Examples

In[187]:=

G = 

In[188]:=

```
PropertyList[{G, 1}]
```

Out[188]=

```
{Race, VertexCoordinates, VertexShape,
 VertexShapeFunction, VertexSize, VertexStyle}
```

Another common name for homophily: Assortative Mixing

(Assortative mating is a nonrandom mating pattern in which individuals with similar genotypes and/or phenotypes mate with one another more frequently than would be expected under a random mating pattern. For example, it is common for individuals of similar body size to mate with one another.)

Measuring Homophily (Assortative Mixing) - Take 2

Enumerative Characteristics

m - number of edges

A_{ij} - Entry in Adjacency Matrix

c_i - Class ("type") of node i

$\delta(c_i, c_j)$ – Kronecker delta function - $\delta(c_i, c_j)$=1 iff c_i=c_j

k_i - degree of node i

Number of edges between vertices of the same type

$$(1) \quad \sum_{\text{edges }(i,j)} \delta(c_i, c_j) = \frac{1}{2}\sum_{ij} A_{ij}\,\delta(c_i, c_j)$$

(why 1/2 ?)

The expected number of random edges between i and j is (why?)

$$\frac{k_i\,k_j}{2\,m}$$

The expected number of random edges between all pairs of vertices of the same type is

$$(2) \quad \frac{1}{2}\sum_{ij}\frac{k_i\,k_j}{2\,m}\,\delta(c_i, c_j)$$

The difference between (1) and (2) is the difference between the expected and the observed number of edges

$$(3) \quad \frac{1}{2}\sum_{ij} A_{ij}\,\delta(c_i, c_j) - \frac{1}{2}\sum_{ij}\frac{k_i\,k_j}{2\,m}\,\delta(c_i, c_j) =$$

$$\frac{1}{2}\sum_{ij}\left(A_{ij} - \frac{k_i\,k_j}{2\,m}\right)\delta(c_i, c_j)$$

This is the number of edges, if we want the fraction of edges we divide by m

$$Q = \frac{1}{2\,m}\sum_{ij}\left(A_{ij} - \frac{k_i\,k_j}{2\,m}\right)\delta(c_i, c_j)$$

This is called the modularity and is a measure of the extent to which like is connected to like in a network.

Q is always Less or equal to 1. But to normalize between networks we divide by the maximum value possible for the network.

$$Q_{max} = \frac{1}{2\,m}\left(2\,m - \sum_{ij} \frac{k_i\,k_j}{2\,m}\,\delta\,(c_i,\,c_j)\right)$$

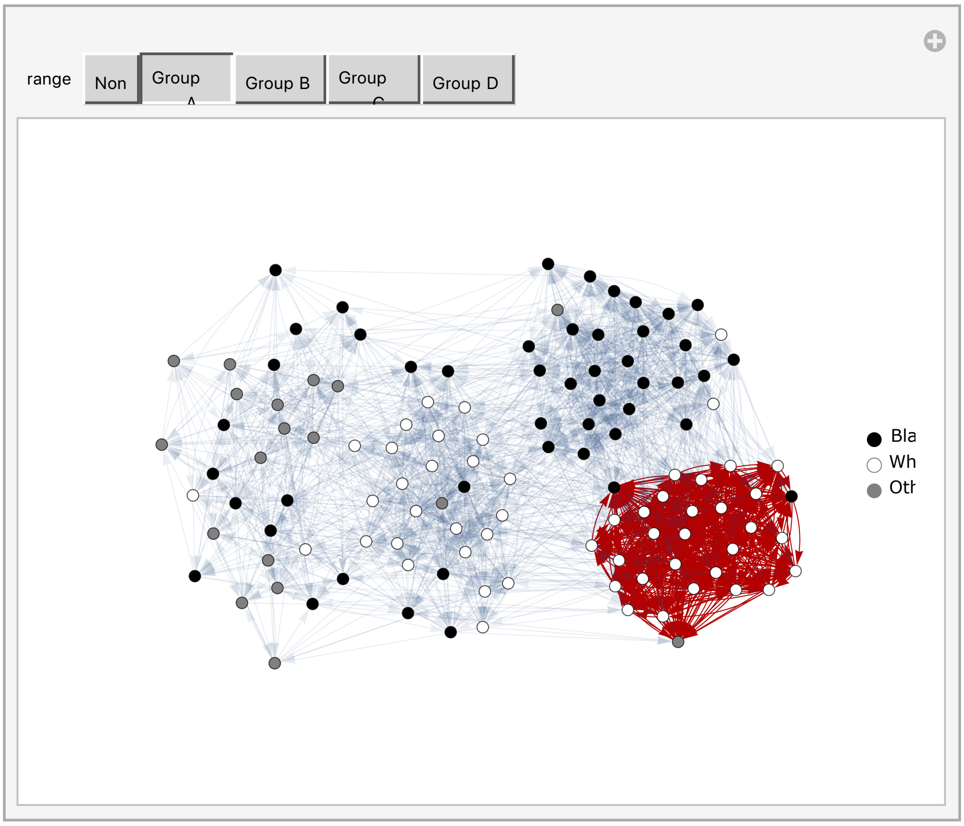
Q/Q_{max} = 1 is a strong positive assortative mixing

Q/Q_{max} = -1 is a strong negative assortative mixing

```
N[GraphAssortativity[G, "Race"]]
```

0.27094

```
Manipulate[HighlightGraph[G, EdgeList[Subgraph[G, range]]],
 {range, {{} → "Non", Range[30] → "Group A", Range[31, 60] → "Group B",
   Range[61, 90] → "Group C", Range[91, 120] → "Group D"}}]
```

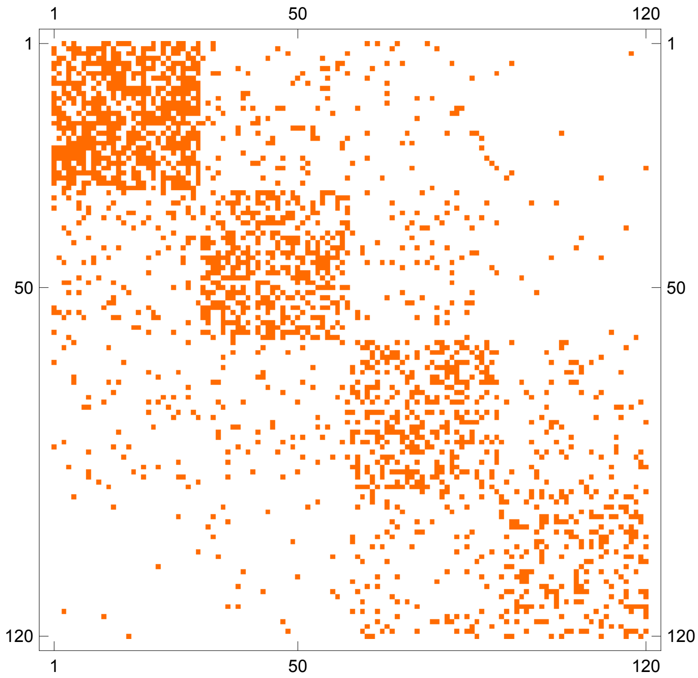


- The Adjacency Matrix may have a structure (like in this example) due to node name/order

In[194]:=

```
MatrixPlot[AdjacencyMatrix[G]]
```

Out[194]=

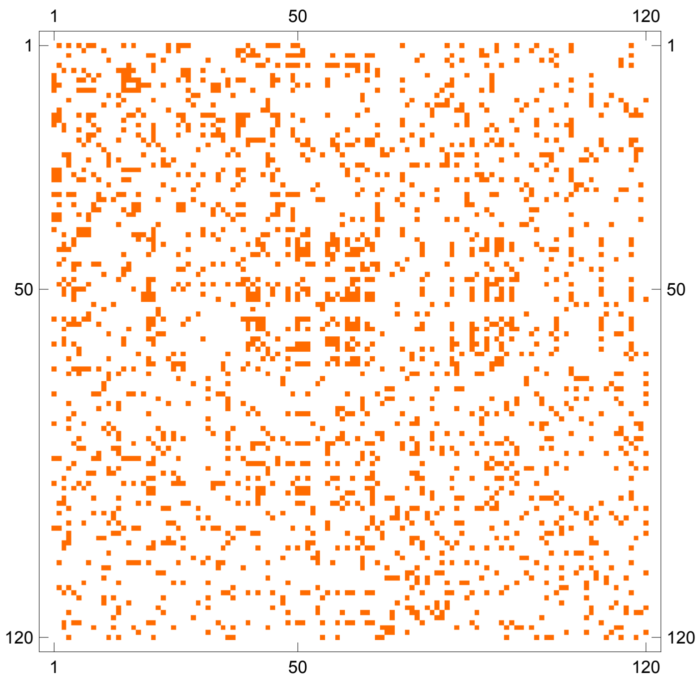


- But usually the node name are disordered (random) so it is hard to observe the structure

In[195]:=

```
MatrixPlot[AdjacencyMatrix[Graph[RandomSample[EdgeList[G], EdgeCount[G]]]]]
```

Out[195]=



- Here is the order by "Race"

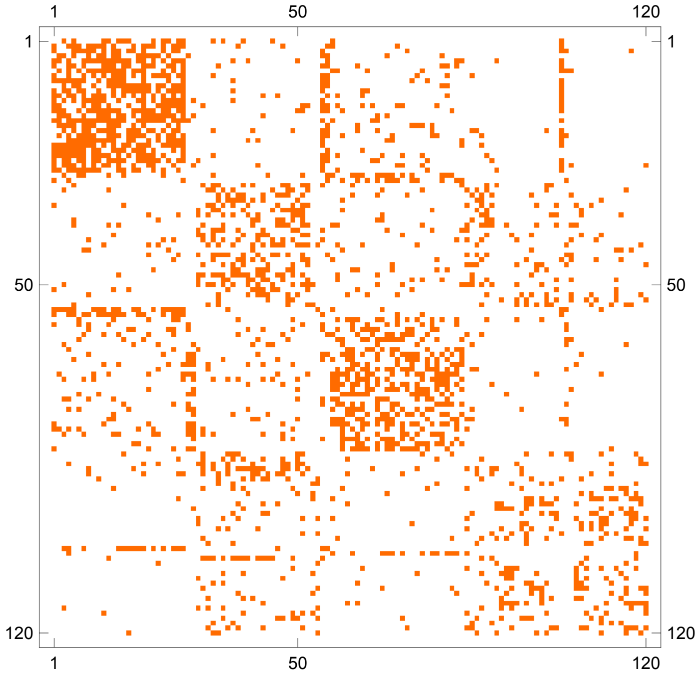
In[196]:=

```
RaceList[race_] := VertexList[G, _?(PropertyValue[{G, #}, "Race"] == race &)];
```

In[197]:=

```
MatrixPlot[AdjacencyMatrix[
  Graph[Flatten[{RaceList[□], RaceList[■], RaceList[■]}], EdgeList[G]]]]
```

Out[197]=



Real Example - Nationality

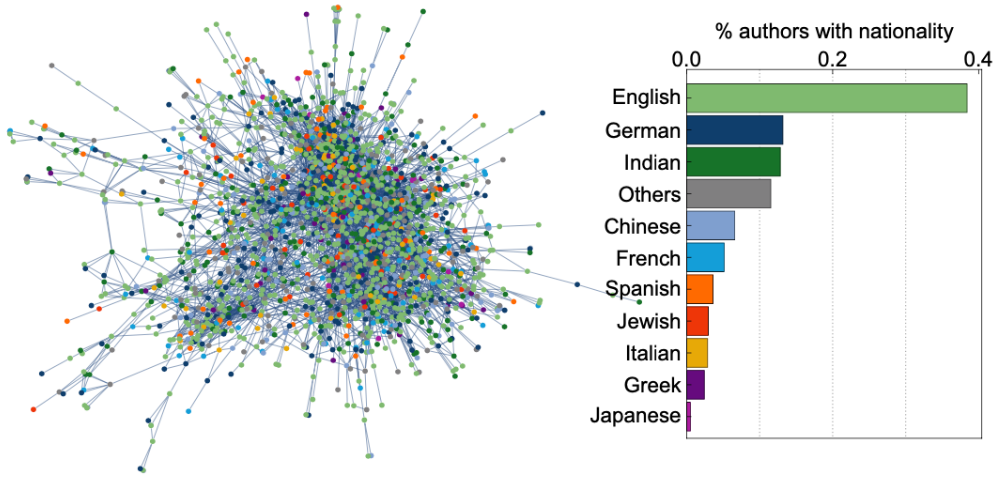


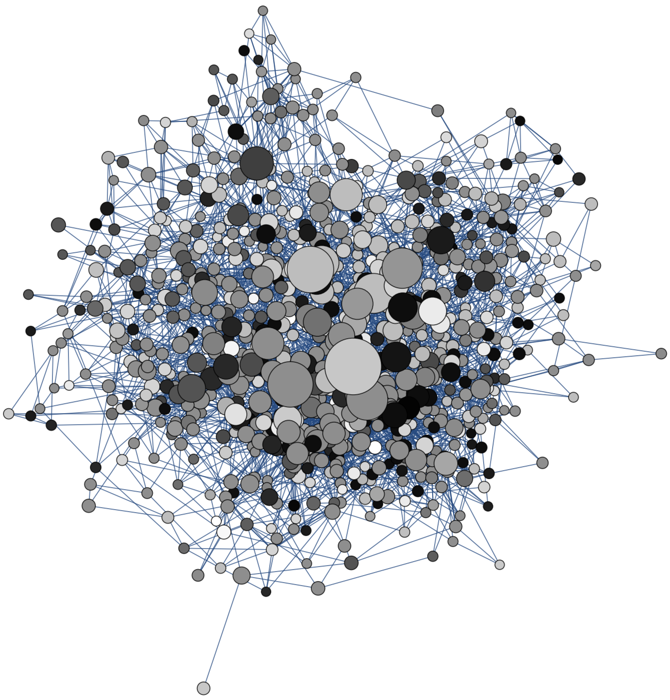
Fig. 1. The top-2000 cited authors social network ($G_{s,u}$). The colors of the nodes correspond to their nationalities.

A graph of the top 800 Computer Science researchers. Edges are by collaboration. Color is by nationality.

In[200]:=

```
SetDirectory[NotebookDirectory[]];
dblpnations = Import["dblptopnations.gxl"]
```

Out[201]=



In[229]:=

```
PropertyValue[dblpnations, VertexStyle];
```

In[225]:=

```
PropertyList[dblpnations]
```

Out[225]=

```
{GraphHighlight, GraphHighlightStyle, GraphLayout,
 GraphStyle, EdgeShapeFunction, EdgeStyle, VertexCoordinates,
 VertexShapeFunction, VertexShape, VertexSize, VertexStyle}
```

In[246]:=

```
PropertyValue[{dblpnations, "121978"}, VertexStyle]
```

Out[246]=

- Assortativity by node color (nationality)

In[202]:=

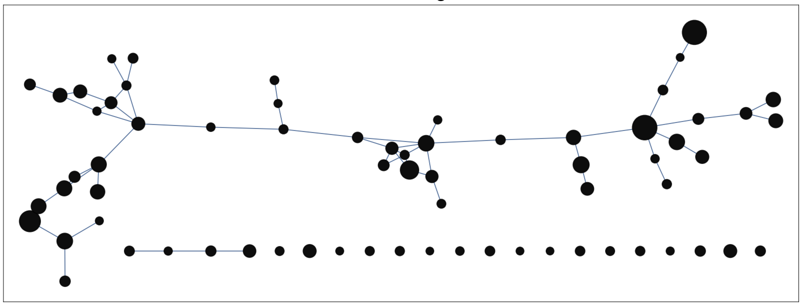
```
N[GraphAssortativity[dblpnations, VertexStyle]]
```

Out[202]=

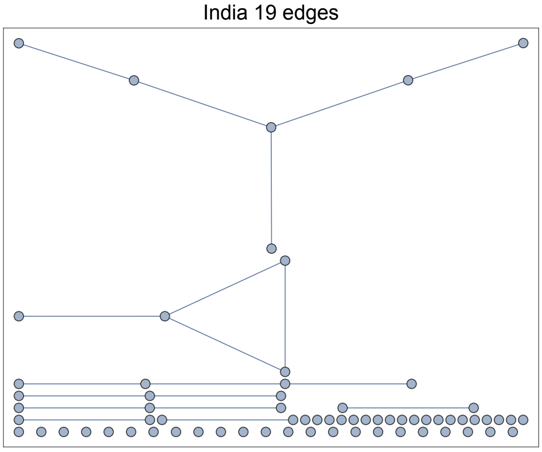
```
0.0691226
```

- Let' s compare some nations to random coloring

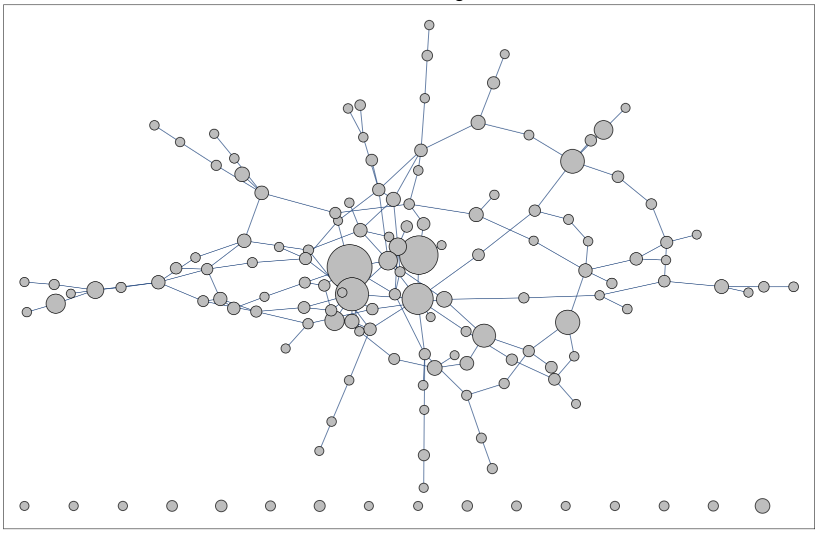
India 57 edges



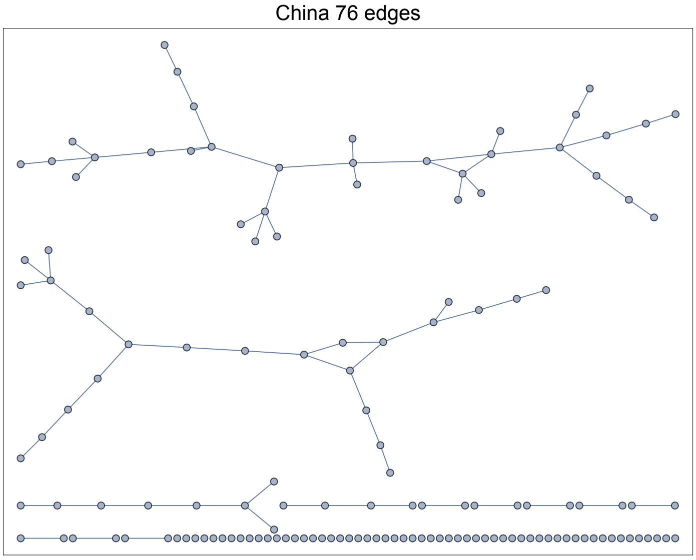
India 19 edges



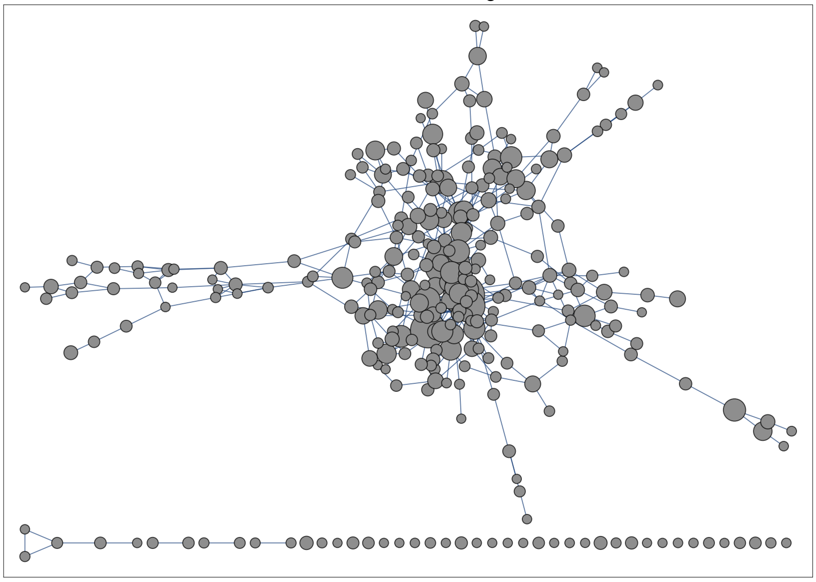
China 160 edges



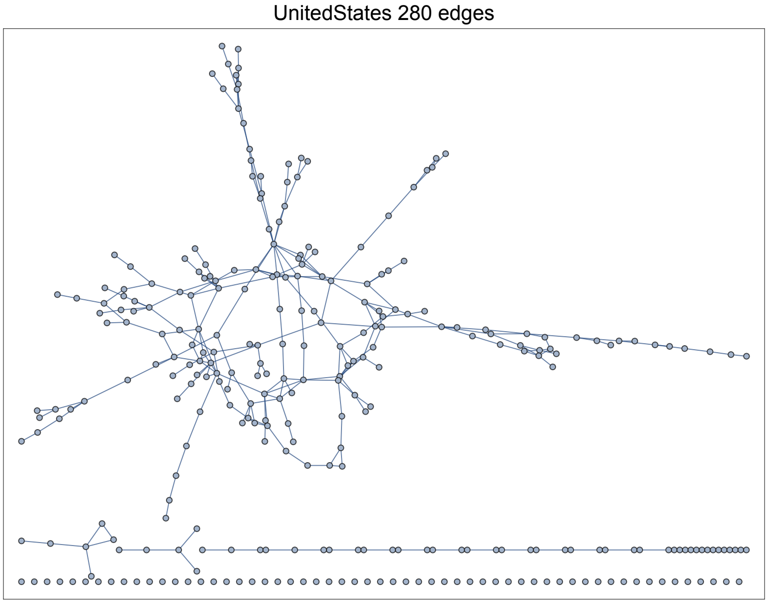
China 76 edges



UnitedStates 504 edges



UnitedStates 280 edges

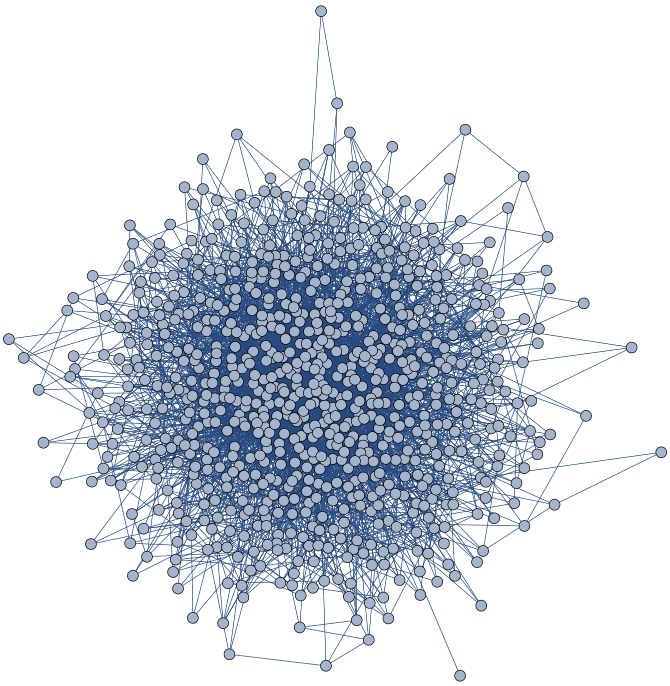


- Random configuration model

In[235]:=

```
CM = RandomGraph[DegreeGraphDistribution[VertexDegree[dblpnations]]]
```

Out[235]=



In[238]:=

```
VertexCount[CM]
```

Out[238]=

```
800
```

In[260]:=

```
NodeStyle = PropertyValue[dblpnations, VertexStyle][[All, 2]];
```

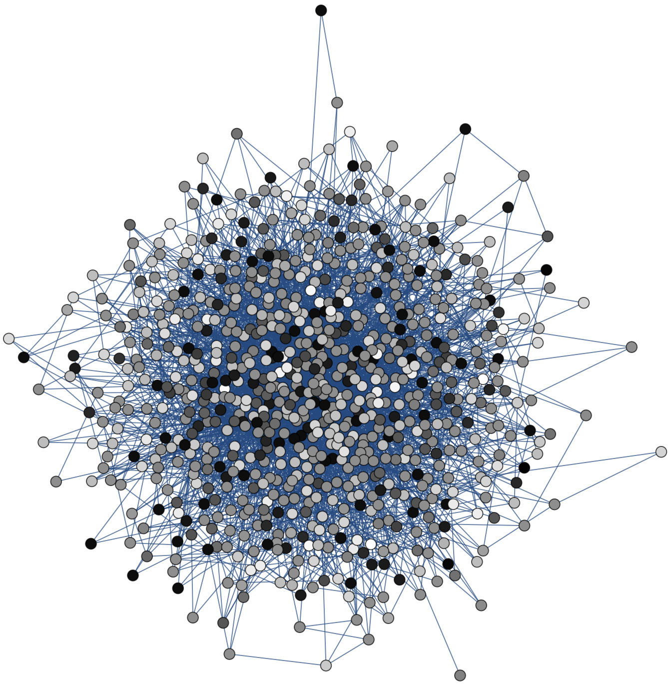
In[277]:=

```
Table[i → NodeStyle[[i]], {i, 800}];
```

In[268]:=

```
TT = Graph[Range[800], EdgeList[CM],
  VertexStyle → Table[i → NodeStyle[[i]], {i, 800}]]
```

Out[268]=



In[275]:=

```
PropertyValue[TT, VertexStyle];
```

In[276]:=

```
PropertyValue[dblpnations, VertexStyle];
```

In[278]:=

```
N[GraphAssortativity[CM, VertexStyle]]
```

Out[278]=

```
0.
```

In[281]:=

```
N[GraphAssortativity[dblpnations, VertexStyle]]
```

Out[281]=

```
0.0691226
```

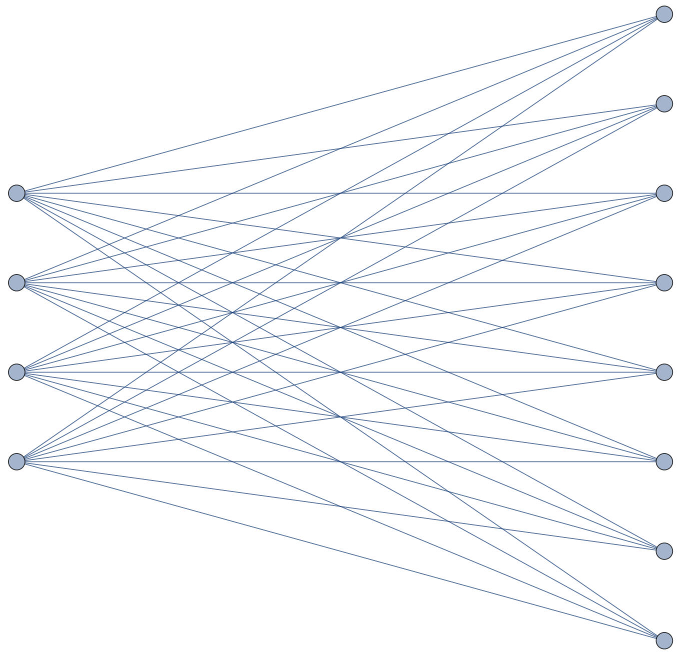
Extreme Example of modularity

Let's see an extreme example of a completed bi-bipartite (700,100) graph in it’s complement

In[203]:=

```
CompleteGraph[{4, 8}]
```

Out[203]=



In[204]:=

```
CG = CompleteGraph[{100, 700}];
```

In[205]:=

```
GraphAssortativity[CG, {Range[100], Range[101, 700]}]
```

Out[205]=

```
-1
```

In[206]:=

```
GraphAssortativity[GraphComplement[CG], {Range[100], Range[101, 700]}]
```

Out[206]=

```
1
```

Scalar Characteristics (age, income, degree, etc.)

- We can group vertices into bins and use each bin as a “type”
- This may be too crude. All vertices in the same bin are considered the “same” and all vertices in other bins are “different”.
- Want to do it using scalar characteristics.

x_i - The value for vertex i of the scalar quantity (age, income, etc.)

Let μ be the mean of x_i at the end of the edges (note: not over nodes)

$$\mu = \frac{\sum_{ij} A_{ij}\, x_i}{\sum_{ij} A_{ij}} = \frac{\sum_i k_i\, x_i}{\sum_i k_i} = \frac{1}{2\,m} \sum_i k_i\, x_i$$

The basic idea is to use covariance (a measure of how much two random variables change together) to determine the relation between x_i to x_j. (Can think of X as a vector of values at the end of edges).

The covariance of x_i and x_j over edges is:

$$\begin{aligned}
\mathrm{cov}\,(x_i, x_j) &= \frac{\sum_{ij} A_{ij}\,(x_i - \mu)\,(x_j - \mu)}{\sum_{ij} A_{ij}} \\
&= \frac{1}{2\,m} \sum_{ij} A_{ij}\,\left(x_i\, x_j - \mu x_i - \mu x_j + \mu^2\right) \\
&= \frac{1}{2\,m} \sum_{ij} A_{ij}\,\left(x_i\, x_j - \mu^2\right) \\
&= \frac{1}{2\,m} \sum_{ij} A_{ij}\, x_i\, x_j - \frac{1}{(2\,m)^2} \sum_{ij} k_i\, k_j\, x_i\, x_j \\
&= \frac{1}{2\,m} \sum_{ij} \left(A_{ij} - \frac{k_i\, k_j}{2\,m}\right) x_i\, x_j
\end{aligned}$$

Very similar to modularity, we replaced $\delta\,(c_i, c_j)$ with $x_i\, x_j$

To normalized between 1 and -1 we can divide by the maximum value. The assortativity coefficient is then

$$(4)\quad r = \frac{\frac{1}{2\,m} \sum_{ij} \left(A_{ij} - \frac{k_i\, k_j}{2\,m}\right) x_i\, x_j}{\frac{1}{2\,m} \sum_{ij} \left(k_i\, \delta_{ij} - \frac{k_i\, k_j}{2\,m}\right) x_i\, x_j}$$

Assortative Mixing by Degree

- Are high-degree nodes prefer to connect to high degree nodes? and low degree to low degree?
- Unlike age or income, degree it a network structure property, here one structural property (degree) dictates another (edge position).
- If high-degree nodes unite together this can lead to a core/periphery structure
- Social networks tends to have positive assortative mixing by degree other networks no.
- Assortative mixing by degree is like any scalar, but replacing x_i with k_i so

$$\text{cov}\,(k_i, k_j) = \frac{1}{2\,m}\sum_{ij}\left(A_{ij} - \frac{k_i\,k_j}{2\,m}\right)k_i\,k_j$$

and (4) become

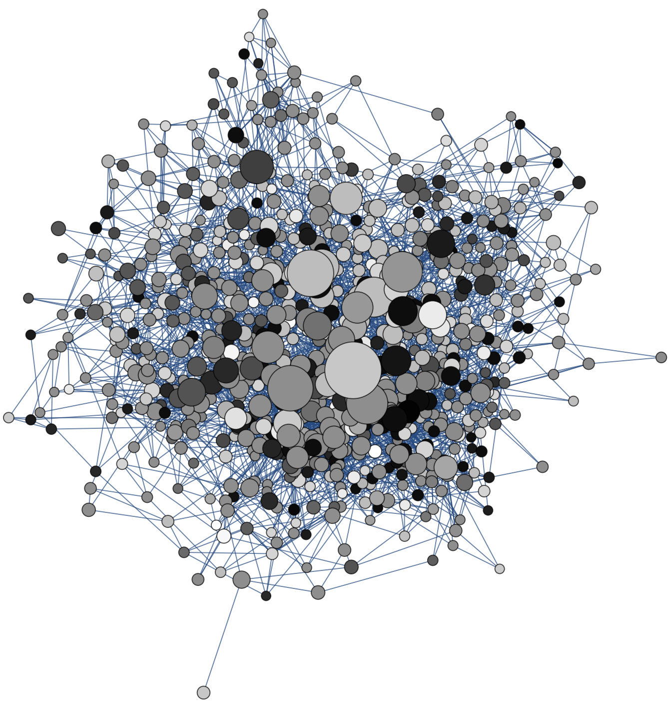
$$r = \frac{\frac{1}{2\,m}\sum_{ij}\left(A_{ij} - \frac{k_i\,k_j}{2\,m}\right)k_i\,k_j}{\frac{1}{2\,m}\sum_{ij}\left(k_i\,\delta_{ij} - \frac{k_i\,k_j}{2\,m}\right)k_i\,k_j}$$

Recall our real example

In[207]:=

```
dblpnations
```

Out[207]=



In[208]:=

```
N[GraphAssortativity[dblpnations, VertexStyle]]
```

Out[208]=

```
0.0691226
```

In[209]:=

```
N[GraphAssortativity[dblpnations]]
```

Out[209]=

```
0.343169
```

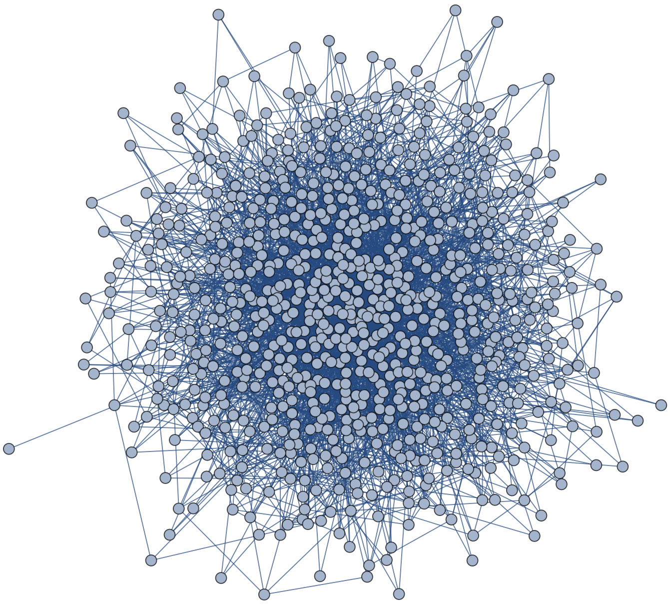
```
Mean[VertexDegree[dblpnations]] // N
```

```
8.78
```

In[210]:=

```
CM = RandomGraph[DegreeGraphDistribution[VertexDegree[dblpnations]]]
```

Out[210]=



In[211]:=

```
GraphAssortativity[CM] // N
```

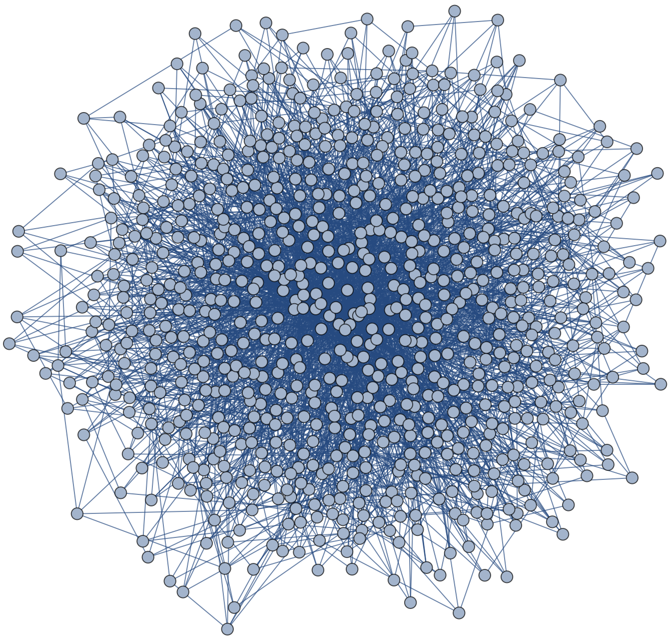
Out[211]=

```
0.00494528
```

In[230]:=

```
R = RandomGraph[BarabasiAlbertGraphDistribution[800, 4]]
```

Out[230]=



In[231]:=

```
Mean[VertexDegree[R]] // N
```

Out[231]=

```
7.975
```

In[232]:=

```
GraphAssortativity[R] // N
```

Out[232]=

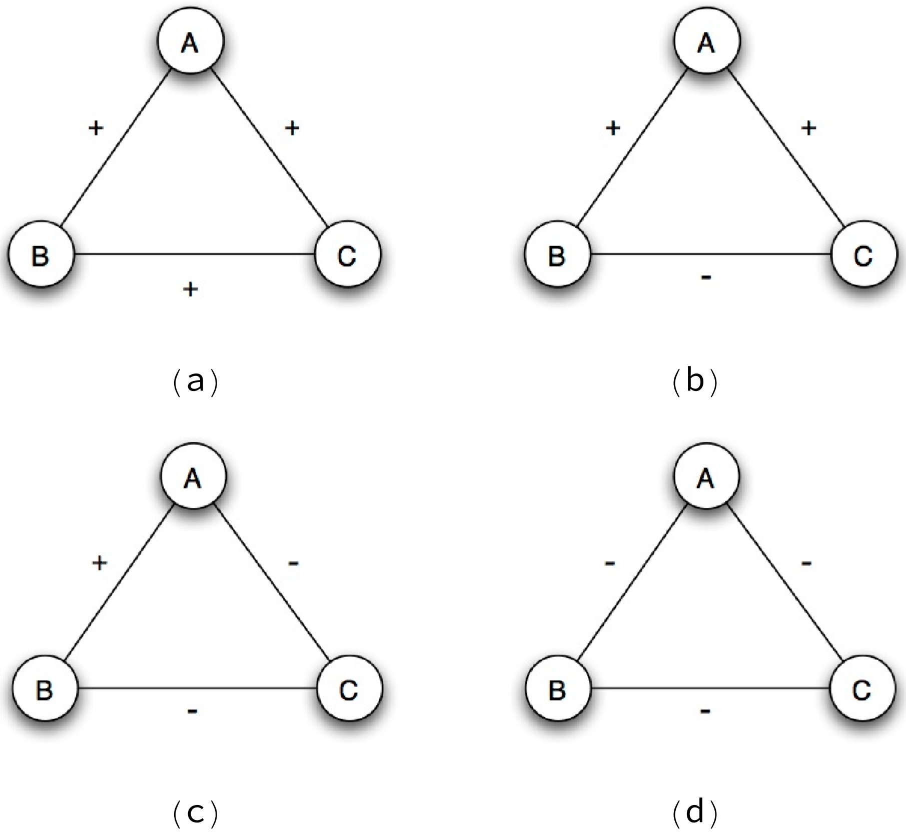
```
-0.0604991
```

Homework 4 - After passover

- Check Modularity for “Network of Thrones” (by 7 tribes) with and without weights.
- Provide examples of 3 "real networks" with positive (away from 0) assortative mixing by degree and 3 "real networks" with negative (away from 0) assortative mixing by degree
- Explain why do you think these networks have this phenomenon
- Find one more network with other scalar/enumerative mixing, positive or negative (e.g., age, income, race, grade, etc.)
- Submit a network with all your code (let me know if you need to upload a network file).

Structural Balance

- Consider both positive and negative relationship
- For now assume a complete graph where every edge is labeled with + or -
- Every pair of people are either friends and enemies (small group, international relationship, etc.)
- Basic idea: For every pair of people the edge between them can be labeled + or -
- But when we look on three people some configuration are socially and psychologically more plausible than others
- Which ones are balanced?



Balanced triangles

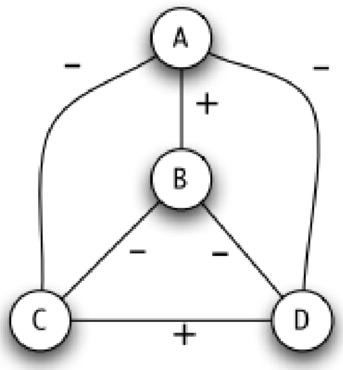
- (a), (c) are balanced (one or three +‘s). Why?
- (b), (d) are not balanced (zero or two +'s). Why?

Balanced Networks

We say that a labeled complete graph is balanced if every one of its triangles is balanced—that is, if it obeys the following :

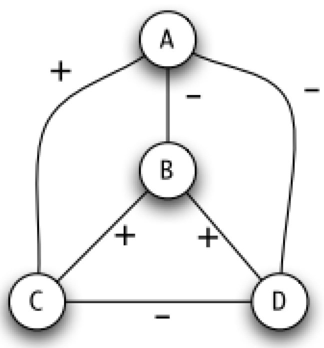
Structural Balance Property *: For every set of three nodes, if we consider the three edges connecting them, either all three of these edges are labeled +, or else exactly one of them is labeled + .*

Balanced Example:



balanced

Not balanced Example:



not balanced

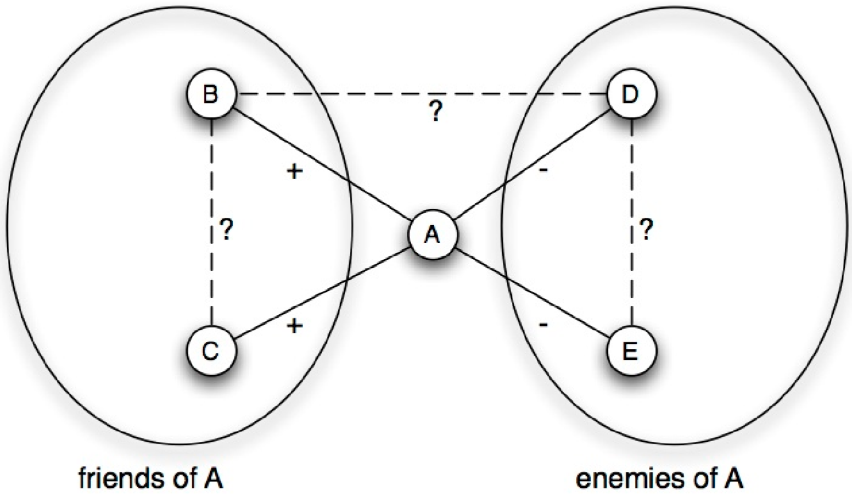
- Easy to check it a network is balanced

But, How does a balanced network looks like?

The Structure of Balanced Networks

The Balance Theorem: *If a labeled complete graph is balanced, then either all pairs of nodes are friends, or else the nodes can be divided into two groups, X and Y, such that every pair of nodes in X like each other, every pair of nodes in Y like each other, and everyone in X is the enemy of everyone in Y.*

- Purely local properly (again) leads to a strong global property
- Proof:
  - Pick any node A in the (balanced) network
  - Let X be the set of all A friends
  - Let Y be the set of all A enemies
  - This is a division of all the network
  - Now, to satisfy the claim, we need to show:
    - (i) Every two nodes in X are friends.
    - (ii) Every two nodes in Y are friends.
    - (ii) Every node in X is an enemy of every node in Y.
  - Let's check the these condition indeed hold...



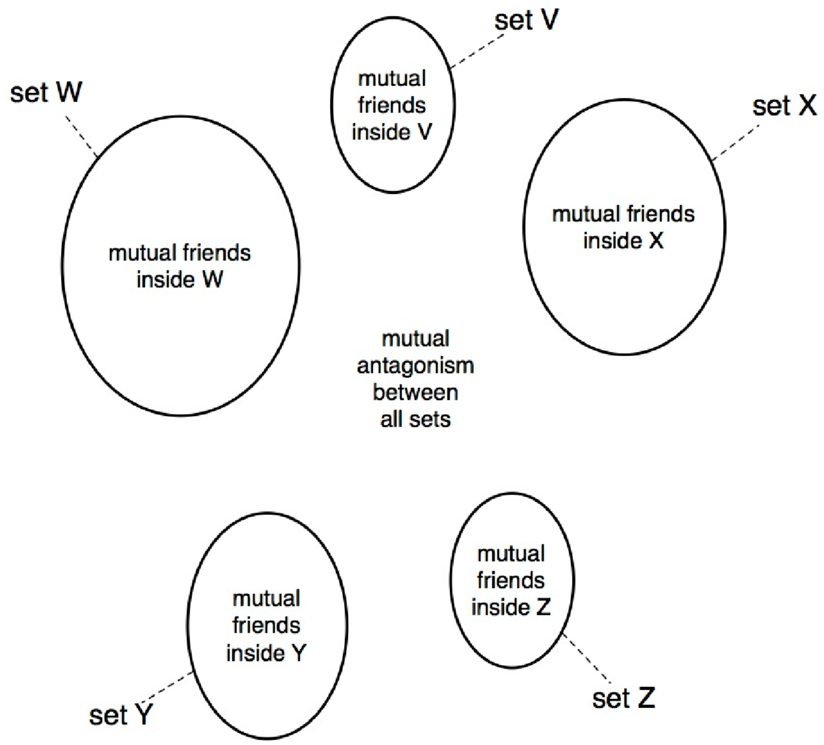
Weakly Balanced Networks

Two unbalanced case (b) and (d).
Claim: case (b) will be resolved since it present a stronger problem. Case (d) may remain.

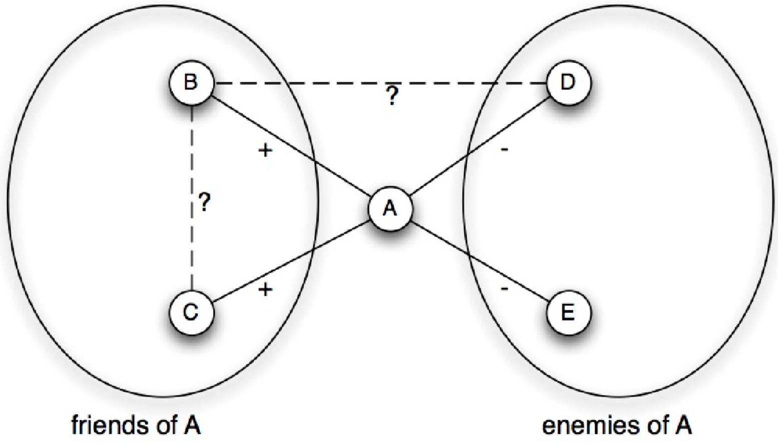
Weak Structural Balance Property *: There is no set of three nodes such that the edges among them consist of exactly two positive edges and one negative edge.*

As a result:

Characterization of Weakly Balanced Networks: If a labeled complete graph is weakly balanced, then its nodes can be divided into groups in such a way that every two nodes belonging to the same group are friends, and every two nodes belonging to different groups are enemies.



- Proof similar to earlier proof, but we need few steps.
- First select a node A.
  - Reason about the neighbors of A.
  - Remove A and the group of its friends and continue



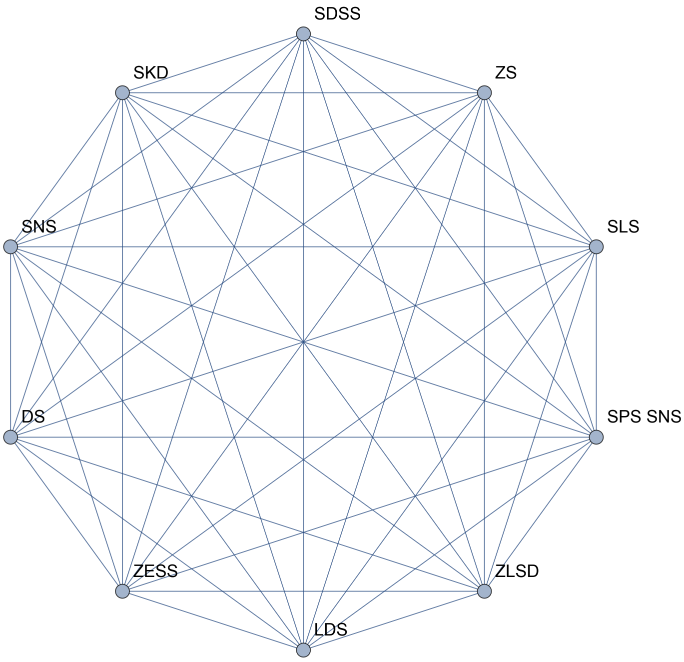
Extensions: i) Balance on non-complete graphs. 2) Graphs that are nearly balanced

Example

In[282]:=

```
sl = ExampleData[{"NetworkGraph", "SloveneParliamentaryParties"}]
```

Out[282]=



In[283]:=

```
WeightedGraphQ[sl]
```

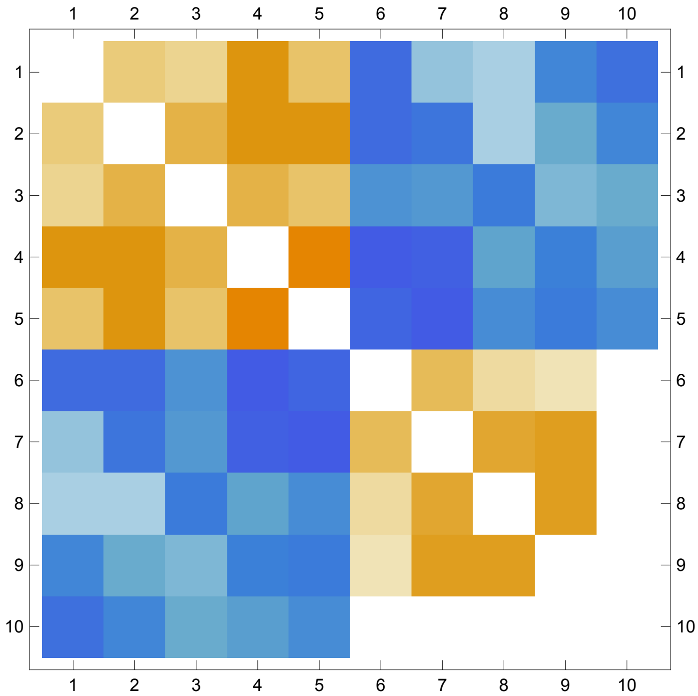
Out[283]=

```
True
```

In[284]:=

```
MatrixPlot[WeightedAdjacencyMatrix[sl]]
```

Out[284]=




```
WeightedAdjacencyMatrix[sl] // MatrixForm
```

$$\begin{pmatrix} 0 & 11 & 9 & 18 & 12 & -22 & -9 & -8 & -17 & -21 \\ 11 & 0 & 14 & 18 & 18 & -22 & -20 & -8 & -11 & -17 \\ 9 & 14 & 0 & 14 & 12 & -15 & -14 & -19 & -10 & -11 \\ 18 & 18 & 14 & 0 & 23 & -25 & -24 & -12 & -18 & -13 \\ 12 & 18 & 12 & 23 & 0 & -23 & -25 & -16 & -19 & -16 \\ -22 & -22 & -15 & -25 & -23 & 0 & 13 & 8 & 6 & 0 \\ -9 & -20 & -14 & -24 & -25 & 13 & 0 & 16 & 17 & 0 \\ -8 & -8 & -19 & -12 & -16 & 8 & 16 & 0 & 17 & 0 \\ -17 & -11 & -10 & -18 & -19 & 6 & 17 & 17 & 0 & 0 \\ -21 & -17 & -11 & -13 & -16 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$