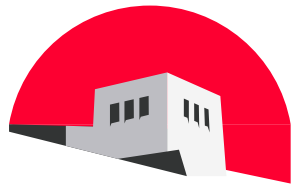
LARGE SCALE BIOLOGICAL NETWORKS

Part I

Complex network theory: An introduction

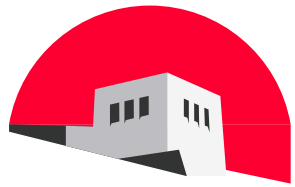
Petter Holme
University of New Mexico, Albuquerque, USA

http://www.cs.unm.edu/~holme/



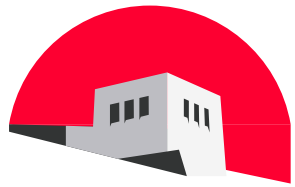
outline

- What systems can be modeled as networks?



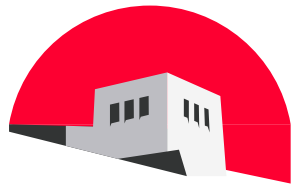
outline

- What systems can be modeled as networks?
- Software and how to make network programs.



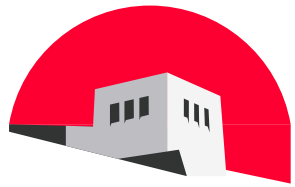
outline

- What systems can be modeled as networks?
- Software and how to make network programs.
- What questions can we ask about a network?



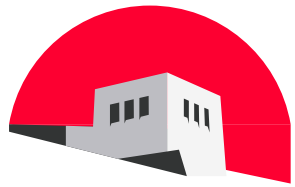
outline

- What systems can be modeled as networks?
- Software and how to make network programs.
- What questions can we ask about a network?
- Measures of network structure.



networked systems

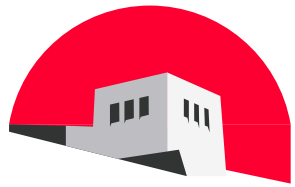
What systems can be modeled as networks?



networked systems

What systems can be modeled as networks?

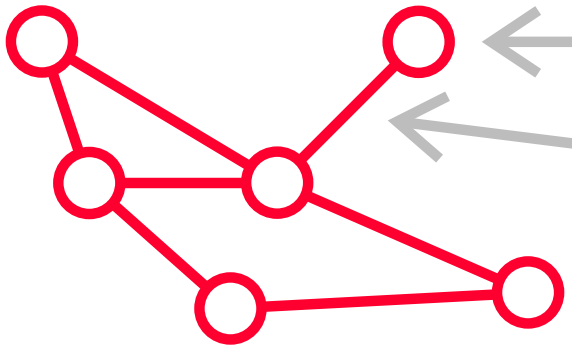
Any system where many objects interact pairwise.



networked systems

What systems can be modeled as networks?

Any system where many objects interact pairwise.



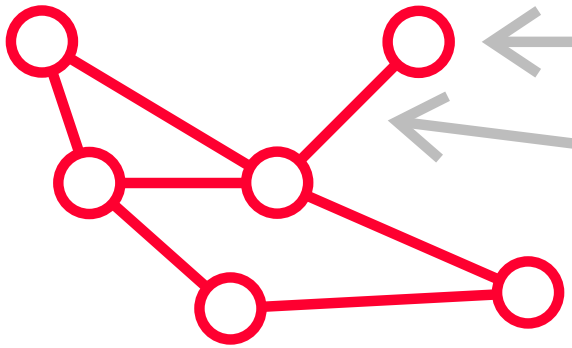
vertex, node, site, actor, agent

edge, link, tie, bond, arc

number of neighbors = *degree*

networked systems

What systems can be modeled as networks?
Any system where many objects interact pairwise.

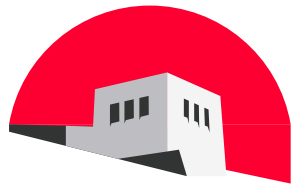


vertex, node, site, actor, agent

edge, link, tie, bond, arc

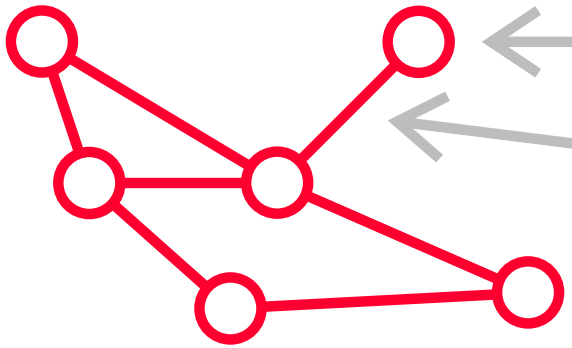
number of neighbors = *degree*

When is network modeling useful?



networked systems

What systems can be modeled as networks?
Any system where many objects interact pairwise.

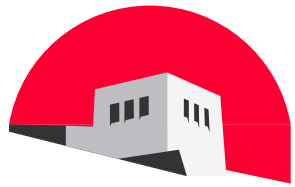


vertex, node, site, actor, agent

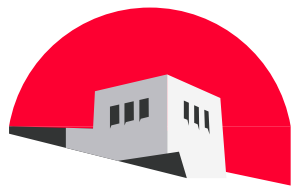
edge, link, tie, bond, arc

number of neighbors = *degree*

When is network modeling useful?
When the average degree is limited.



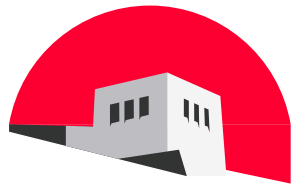
networked systems



networked systems

constructing networks: example

1. Index the names.



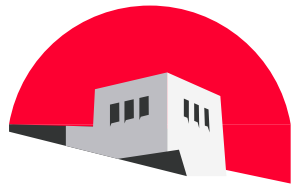
constructing networks: example

1. Index the names.
2. 

constructing networks: example

1. Index the names.
2. 

3. Construct a matrix with the number of hits H_{ij}.



constructing networks: example

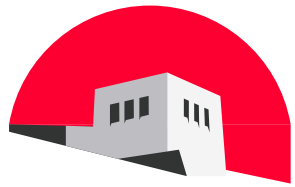
1. Index the names.
2. 

3. Construct a matrix with the number of hits H_{ij}.
4. Let $h_i = \sum_j H_{ij}$.

constructing networks: example

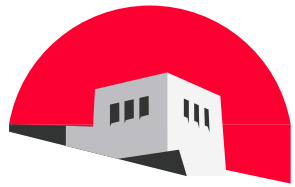
1. Index the names.
2. 

3. Construct a matrix with the number of hits H_{ij}.
4. Let $h_i = \sum_j H_{ij}$.
5. If $w_{ij} = H_{ij} / \sqrt{h_i h_j} > \theta$ for a threshold θ make an edge between i and j.



constructing networks: example

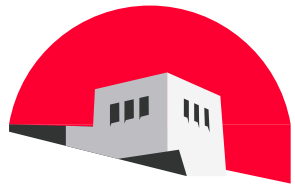
Things to ask oneself:



constructing networks: example

Things to ask oneself:

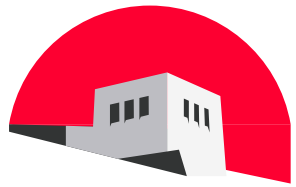
- What is my network, really? What do the edges really represent?



constructing networks: example

Things to ask oneself:

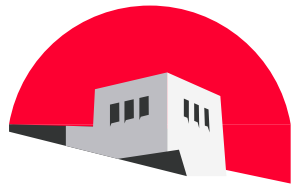
- What is my network, really? What do the edges really represent?
- Are there false edges? Missing edges?



constructing networks: example

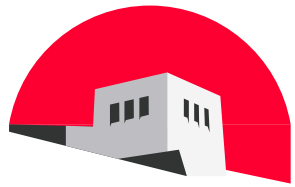
Things to ask oneself:

- What is my network, really? What do the edges really represent?
- Are there false edges? Missing edges?
- How does that affect my result?



analyzing the network

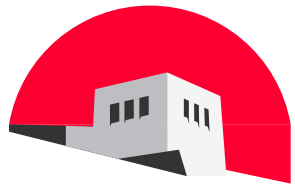
software:



analyzing the network

software:

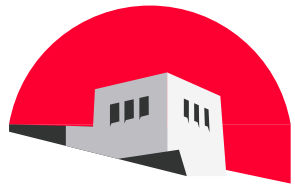
- **Pajek** http://vlado.fmf.uni-lj.si/pub/networks/pajek/ layout + analysis



analyzing the network

software:

- **Pajek** http://vlado.fmf.uni-lj.si/pub/networks/pajek/ layout + analysis
- **UCINET** http://www.analytictech.com/ social networks



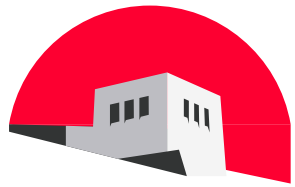
analyzing the network

software:

- **Pajek** http://vlado.fmf.uni-lj.si/pub/networks/pajek/ layout + analysis
- **UCINET** http://www.analytictech.com/ social networks

Java:

- **JUNG** http://jung.sourceforge.net/



analyzing the network

software:

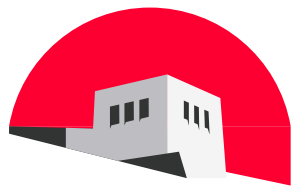
- **Pajek** http://vlado.fmf.uni-lj.si/pub/networks/pajek/ layout + analysis
- **UCINET** http://www.analytictech.com/ social networks

Java:

- **JUNG** http://jung.sourceforge.net/

C++:

- **Boost** http://www.boost.org/ has Python bindings



analyzing the network

software:

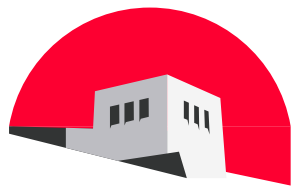
- **Pajek** http://vlado.fmf.uni-lj.si/pub/networks/pajek/ layout + analysis
- **UCINET** http://www.analytictech.com/ social networks

Java:

- **JUNG** http://jung.sourceforge.net/

C++:

- **Boost** http://www.boost.org/ has Python bindings
- **GTL** http://infosun.fmi.uni-passau.de/GTL/



analyzing the network

software:

- **Pajek** http://vlado.fmf.uni-lj.si/pub/networks/pajek/ layout + analysis
- **UCINET** http://www.analytictech.com/ social networks

Java:

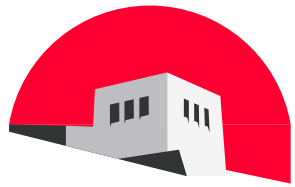
- **JUNG** http://jung.sourceforge.net/

C++:

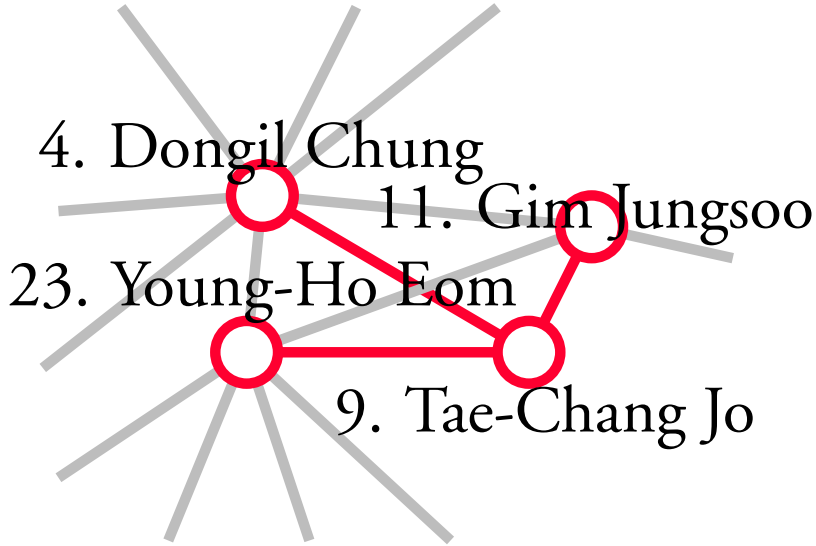
- **Boost** http://www.boost.org/ has Python bindings
- **GTL** http://infosun.fmi.uni-passau.de/GTL/

C, Matlab:

- You have to do most yourself . . .



internal representations

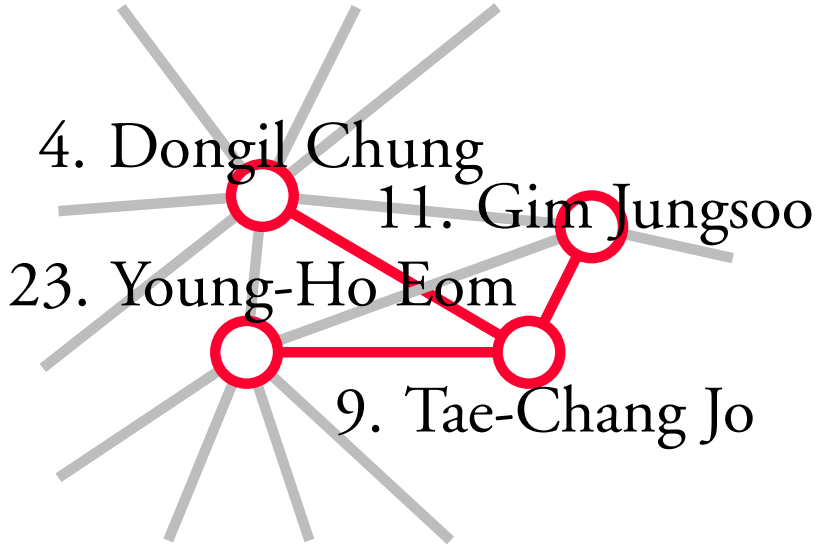


vertex 9, deg 3
neighbors 4, 11, 23

edge 31, one side 9
other side 23

.

internal representations

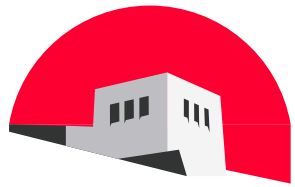


vertex 9, deg 3
neighbors 4, 11, 23

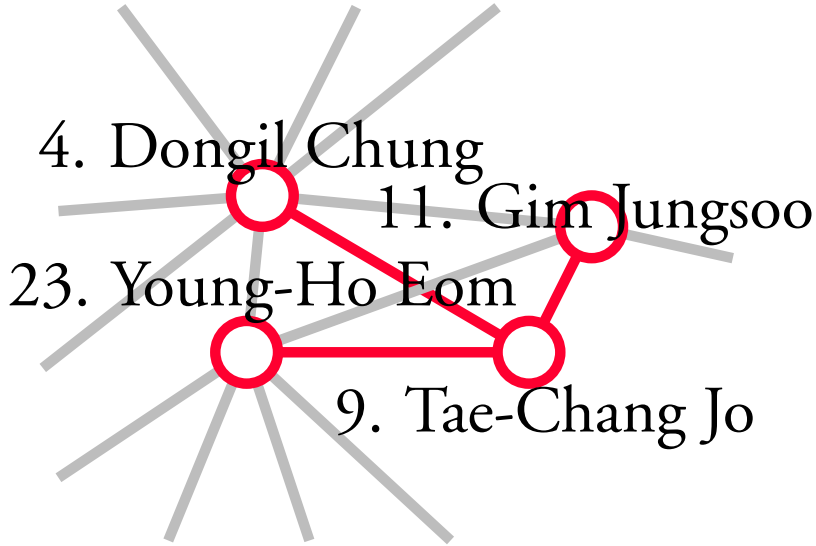
edge 31, one side 9
other side 23

★ For linear algebra methods: matrices.

.



internal representations

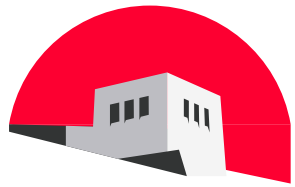


vertex 9, deg 3
neighbors 4, 11, 23

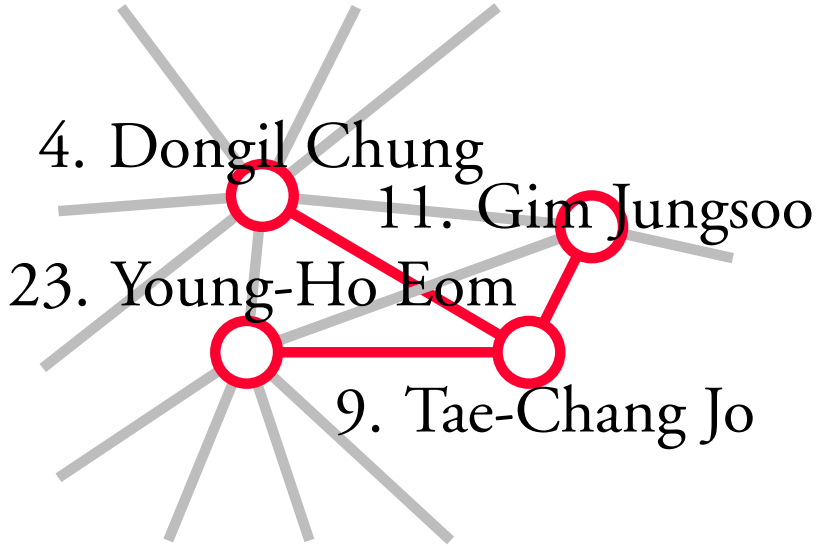
edge 31, one side 9
other side 23

- For linear algebra methods: matrices.
- Otherwise: adjacency lists . . .

.



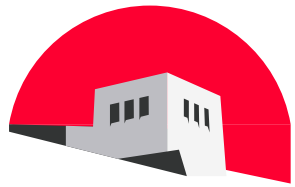
internal representations



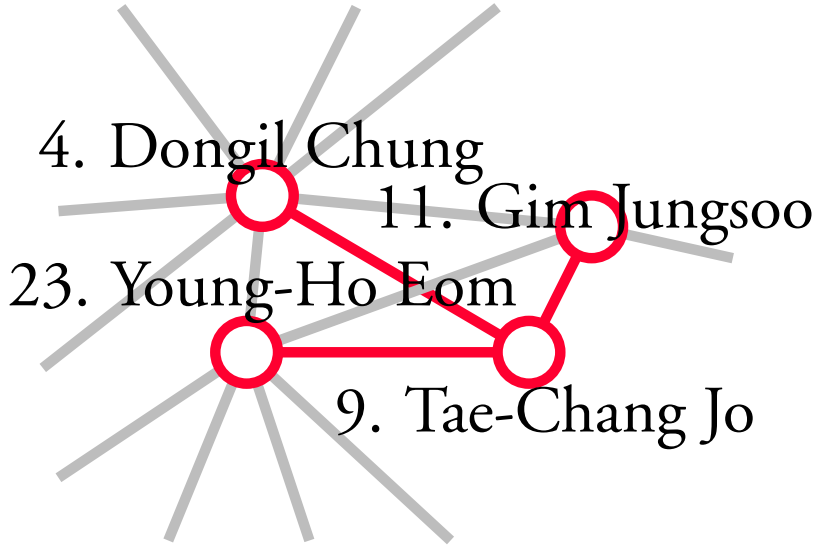
vertex 9, deg 3
neighbors 4, 11, 23

edge 31, one side 9
other side 23

- For linear algebra methods: matrices.
- Otherwise: adjacency lists . . .
- . . . or edge lists .



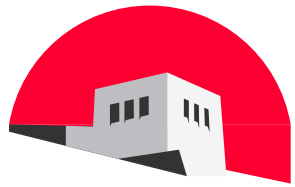
internal representations



vertex 9, deg 3
neighbors 4, 11, 23

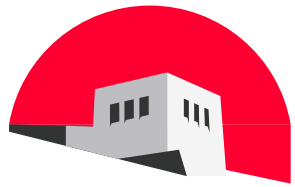
edge 31, one side 9
other side 23

- For linear algebra methods: matrices.
- Otherwise: adjacency lists . . .
- . . . or edge lists (or both).



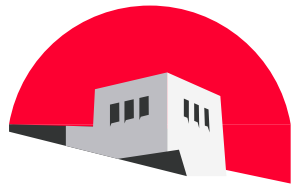
now what?

- We have a system represented as a network.
- We have the tools to analyze it.
- What questions can we ask?



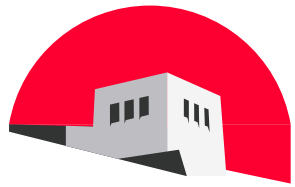
network science

- **Measures of network structure.** How does a network that is too large to draw "look" like? Real-world networks have both randomness and structure. How can we quantify network structure?



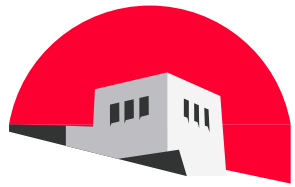
network science

- **Measures of network structure.** How does a network that is too large to draw "look" like? Real-world networks have both randomness and structure. How can we quantify network structure?
- **Models of evolving networks.** How do networks get their structure? What "microscopic" properties are responsible for the macro-structure of the network.



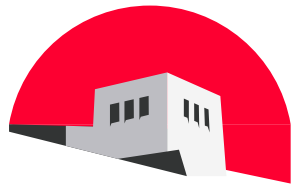
network science

- **Measures of network structure.** How does a network that is too large to draw "look" like? Real-world networks have both randomness and structure. How can we quantify network structure?
- **Models of evolving networks.** How do networks get their structure? What "microscopic" properties are responsible for the macro-structure of the network.
- **Models of network changing events.** Malicious attacks; overload breakdowns.



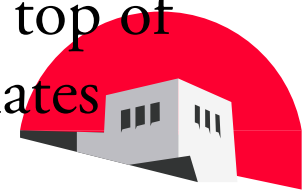
network science

- **Measures of network structure.** How does a network that is too large to draw "look" like? Real-world networks have both randomness and structure. How can we quantify network structure?
- **Models of evolving networks.** How do networks get their structure? What "microscopic" properties are responsible for the macro-structure of the network.
- **Models of network changing events.** Malicious attacks; overload breakdowns.
- **Classification and functional prediction.** How can we classify vertices and predict their function in the network?

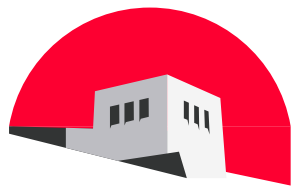


network science

- **Measures of network structure.** How does a network that is too large to draw "look" like? Real-world networks have both randomness and structure. How can we quantify network structure?
- **Models of evolving networks.** How do networks get their structure? What "microscopic" properties are responsible for the macro-structure of the network.
- **Models of network changing events.** Malicious attacks; overload breakdowns.
- **Classification and functional prediction.** How can we classify vertices and predict their function in the network?
- **How does the network structure affect dynamic systems of the network?** Running dynamic simulations on top of the network and see how dynamic properties correlates with the network structure.

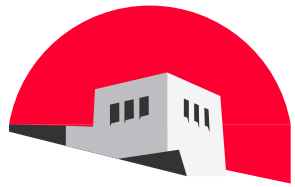


types of structural measures



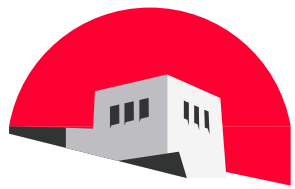
types of structural measures

- Vertex measures.



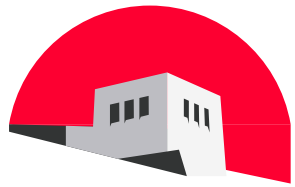
types of structural measures

- Vertex measures.
- Edge measures.



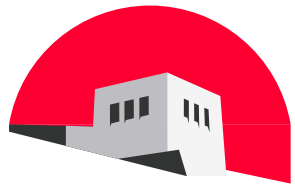
types of structural measures

- Vertex measures.
- Edge measures.
- Vertex-pair measures.



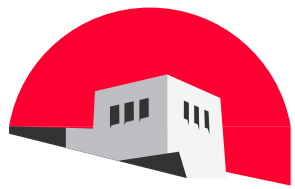
types of structural measures

- Vertex measures.
- Edge measures.
- Vertex-pair measures.
- Graph measures.



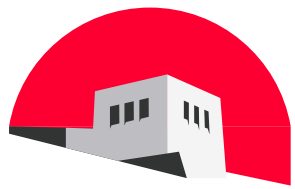
network null-models

- Network structures are always relative . . .



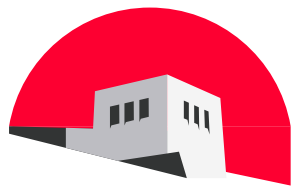
network null-models

- Network structures are always relative . . .
- . . . one has to be clear about what to compare with . . . a null model



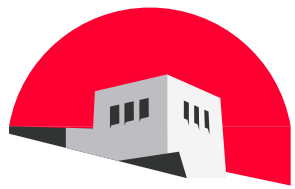
network null-models

- Network structures are always relative . . .
- . . . one has to be clear about what to compare with . . . a null model
- *Null model 1:* random graphs (Poisson random graphs, Erdős-Rényi graphs)

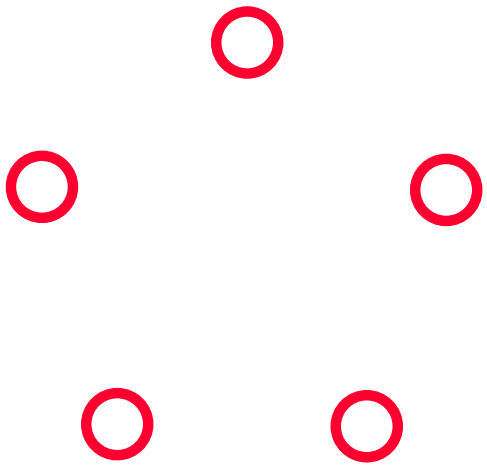


network null-models

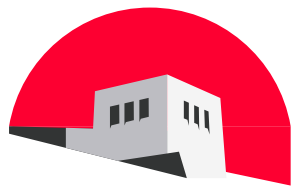
- Network structures are always relative . . .
- . . . one has to be clear about what to compare with . . . a null model
- *Null model 1:* random graphs (Poisson random graphs, Erdős-Rényi graphs)
- *Null model 2:* random graphs constrained to the set of degrees of the original graph



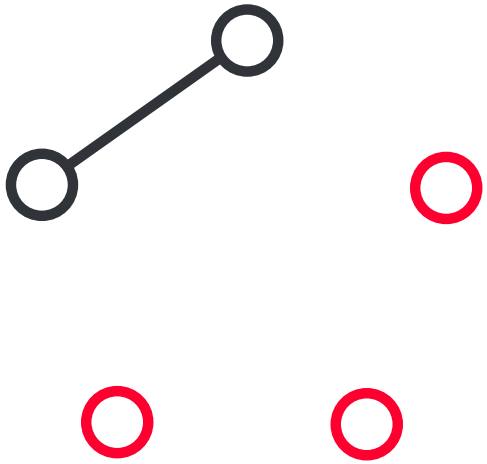
random graphs



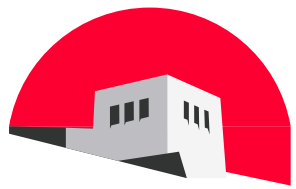
for each pair of vertices,
with probability p, add an edge



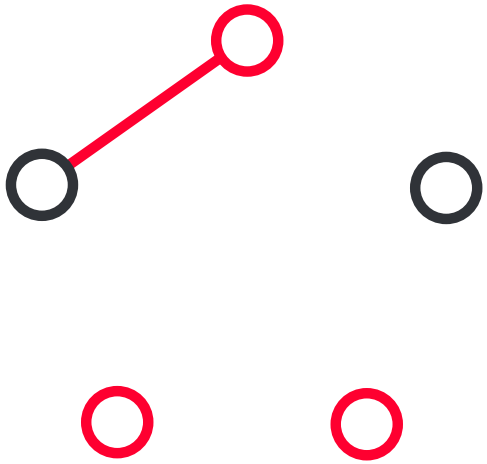
random graphs



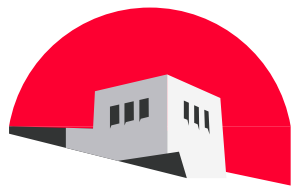
for each pair of vertices,
with probability p, add an edge



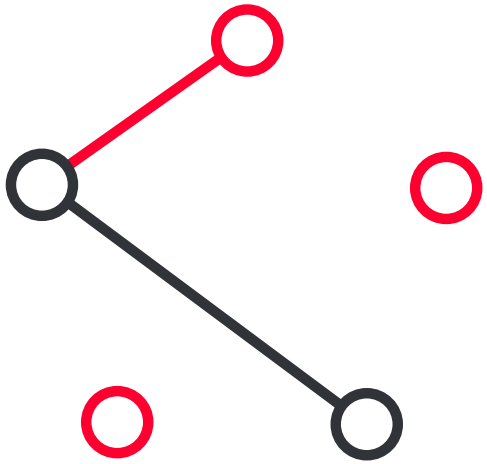
random graphs



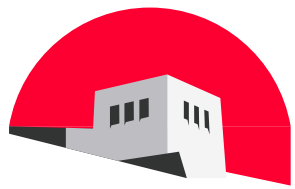
for each pair of vertices,
with probability p, add an edge



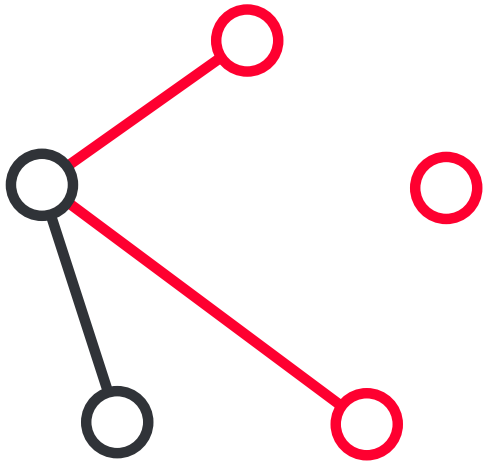
random graphs



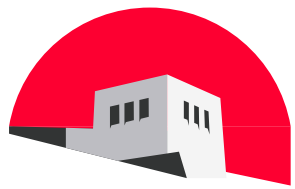
for each pair of vertices,
with probability p, add an edge



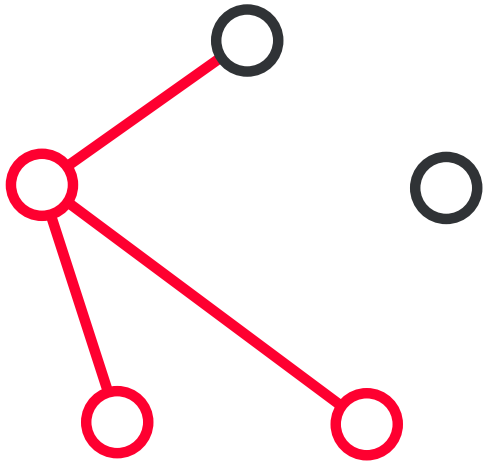
random graphs



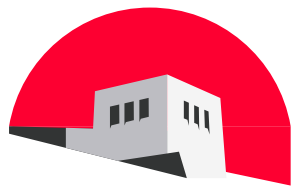
for each pair of vertices,
with probability p, add an edge



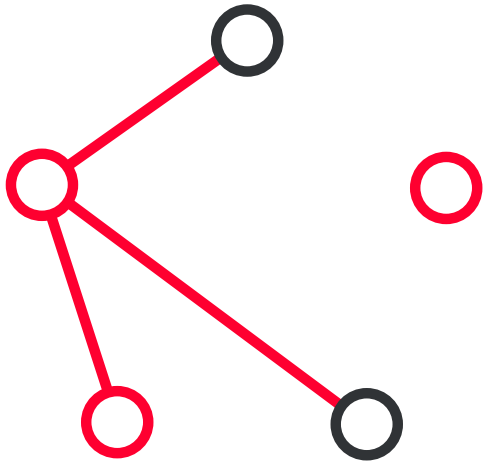
random graphs



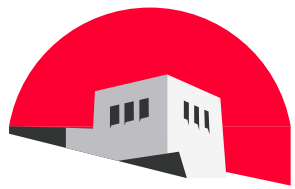
for each pair of vertices,
with probability p, add an edge



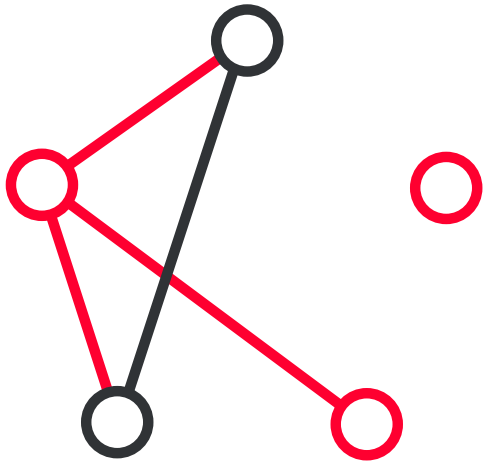
random graphs



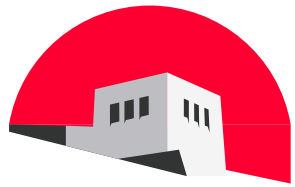
for each pair of vertices,
with probability p, add an edge



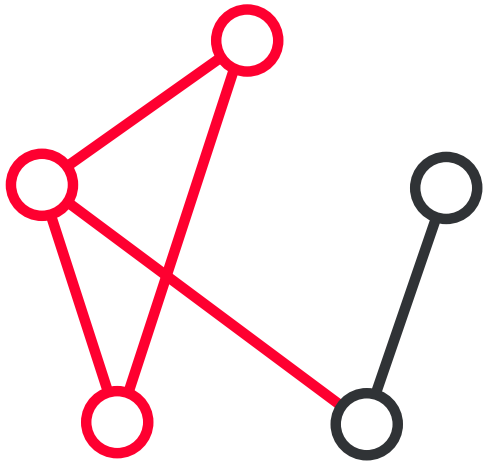
random graphs



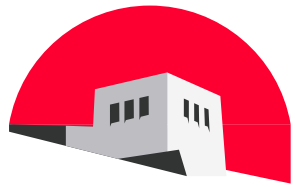
for each pair of vertices,
with probability p, add an edge



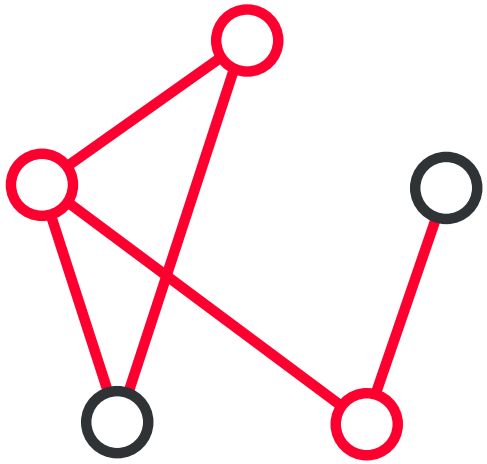
random graphs



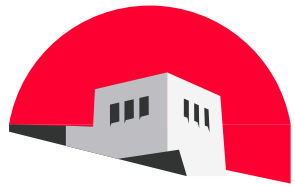
for each pair of vertices,
with probability p, add an edge



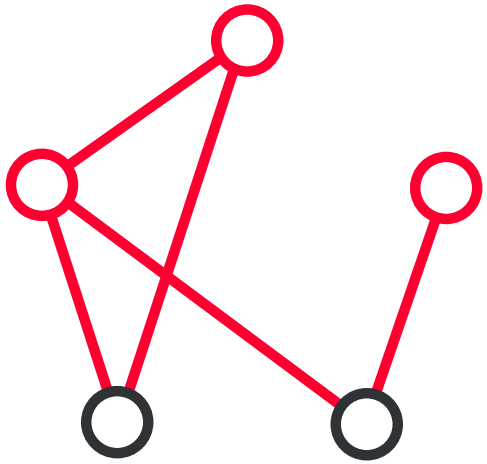
random graphs



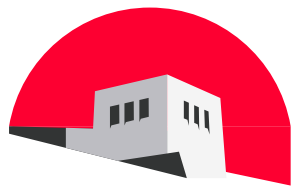
for each pair of vertices,
with probability p, add an edge



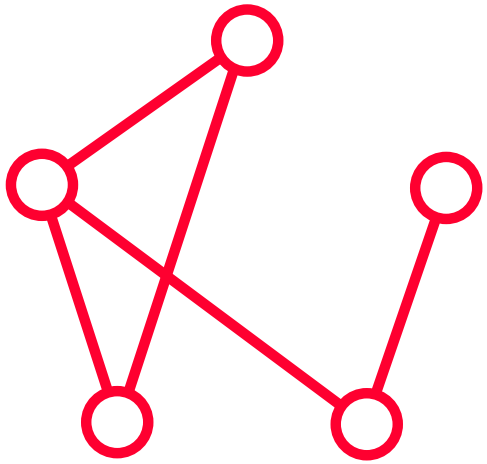
random graphs



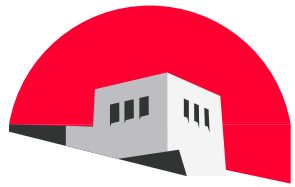
for each pair of vertices,
with probability p, add an edge



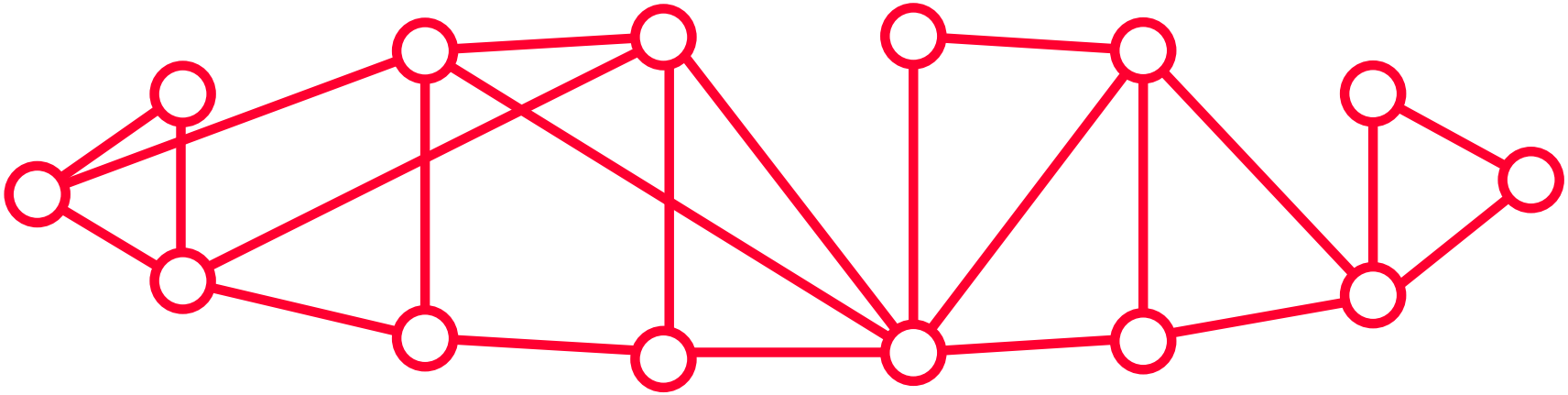
random graphs



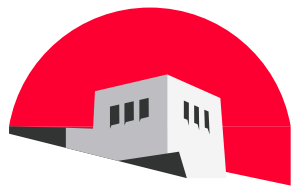
for each pair of vertices,
with probability p, add an edge



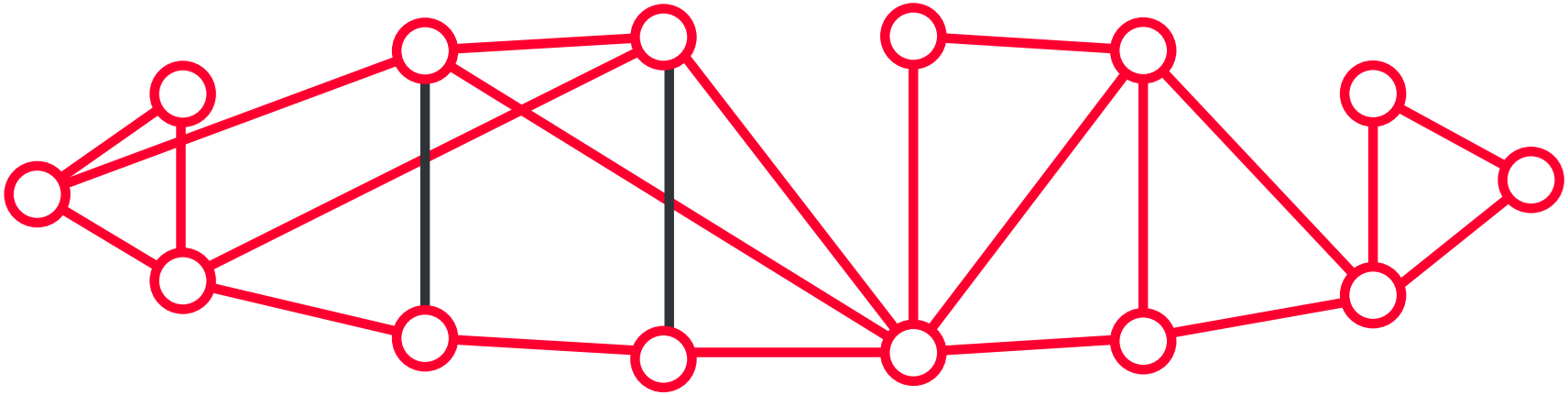
random rewiring



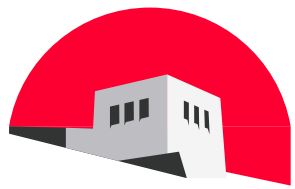
start from the original graph
choose edge pairs, and swap them



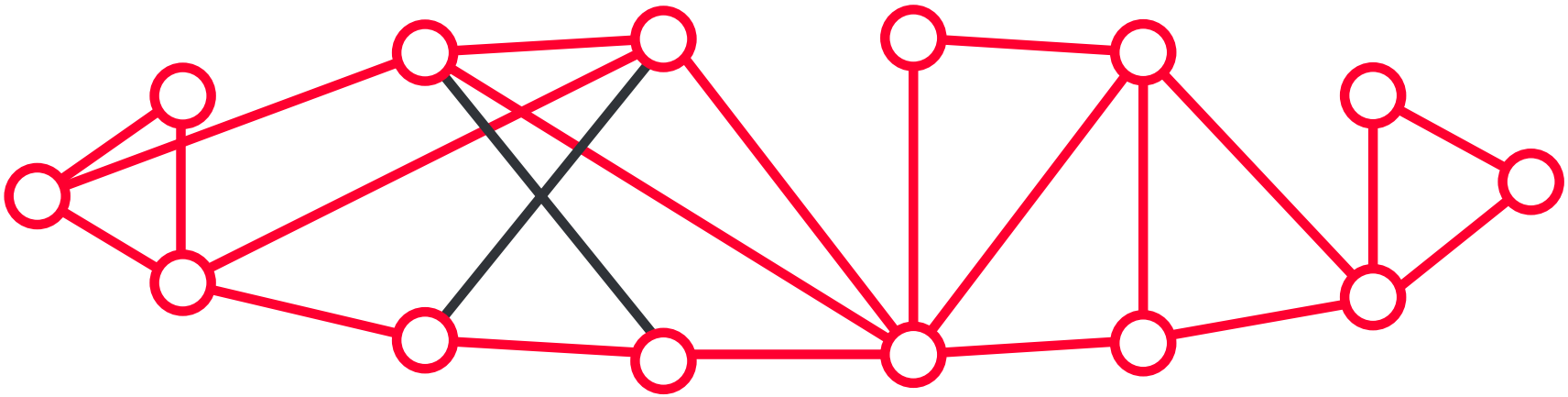
random rewiring



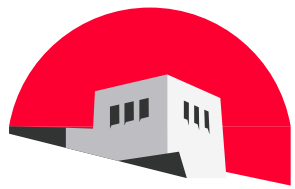
start from the original graph
choose edge pairs, and swap them



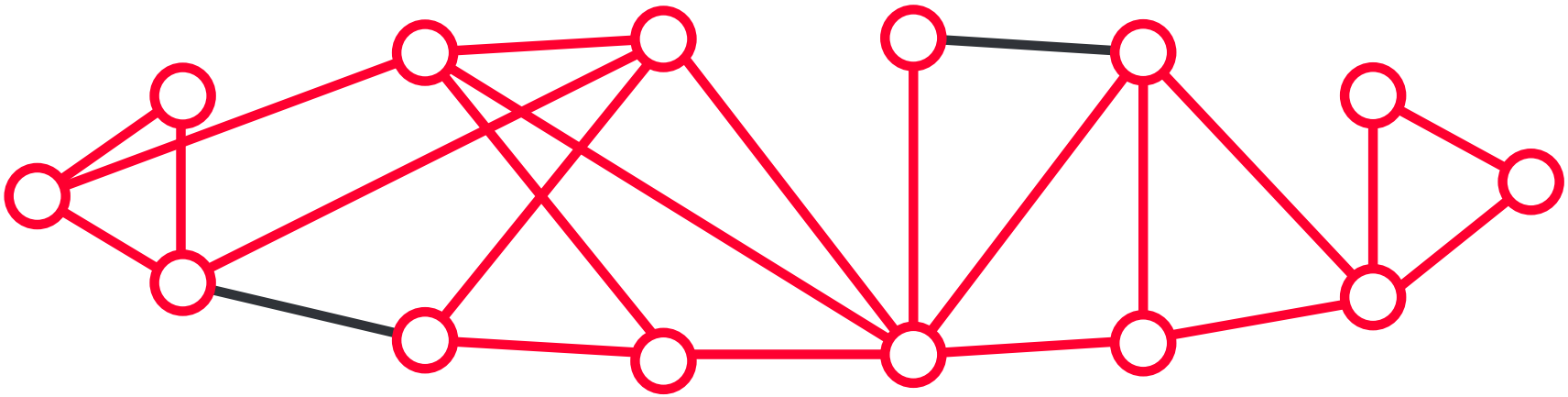
random rewiring



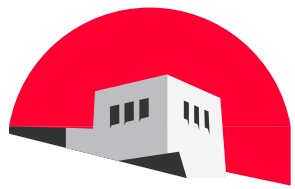
start from the original graph
choose edge pairs, and swap them



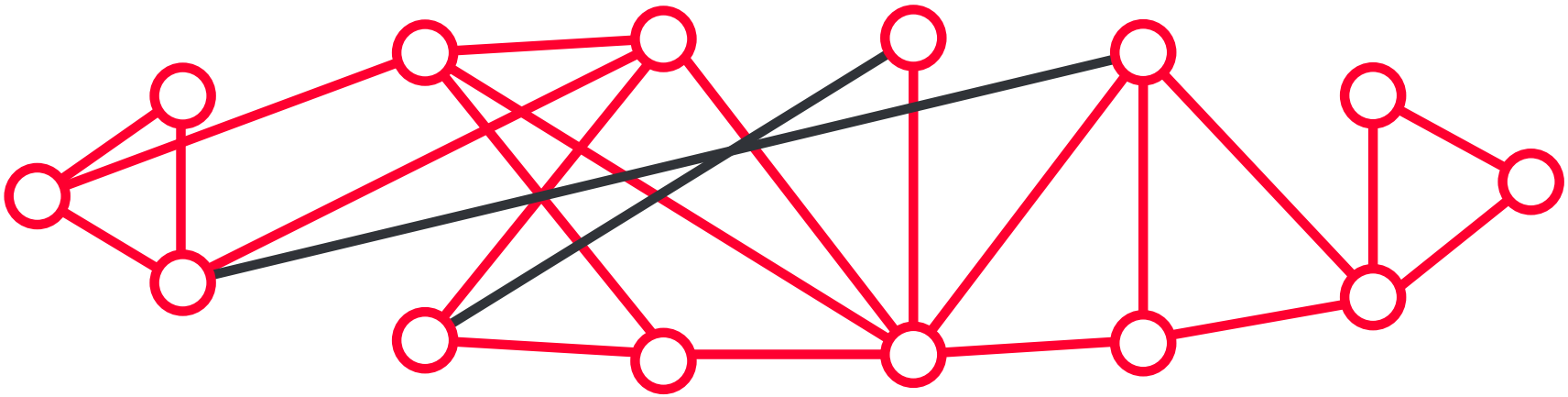
random rewiring



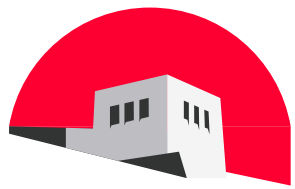
start from the original graph
choose edge pairs, and swap them



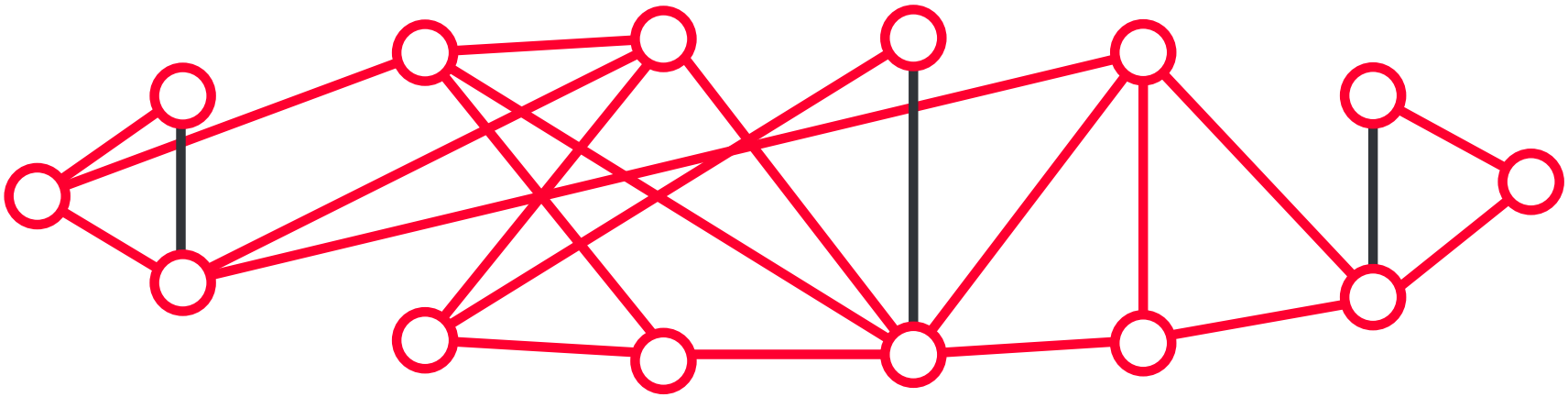
random rewiring



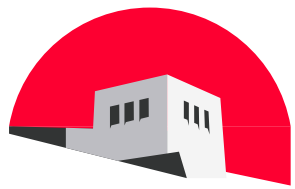
start from the original graph
choose edge pairs, and swap them



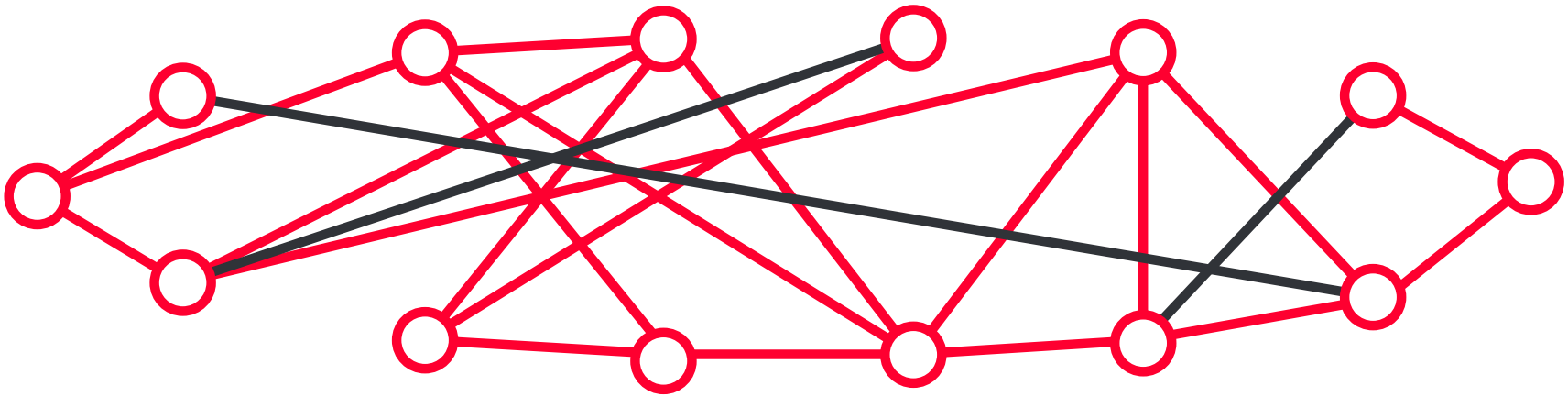
random rewiring



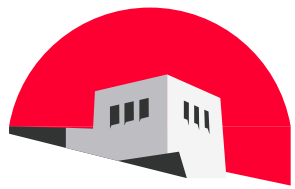
start from the original graph
for ergodicity, edge triples have to be swapped too



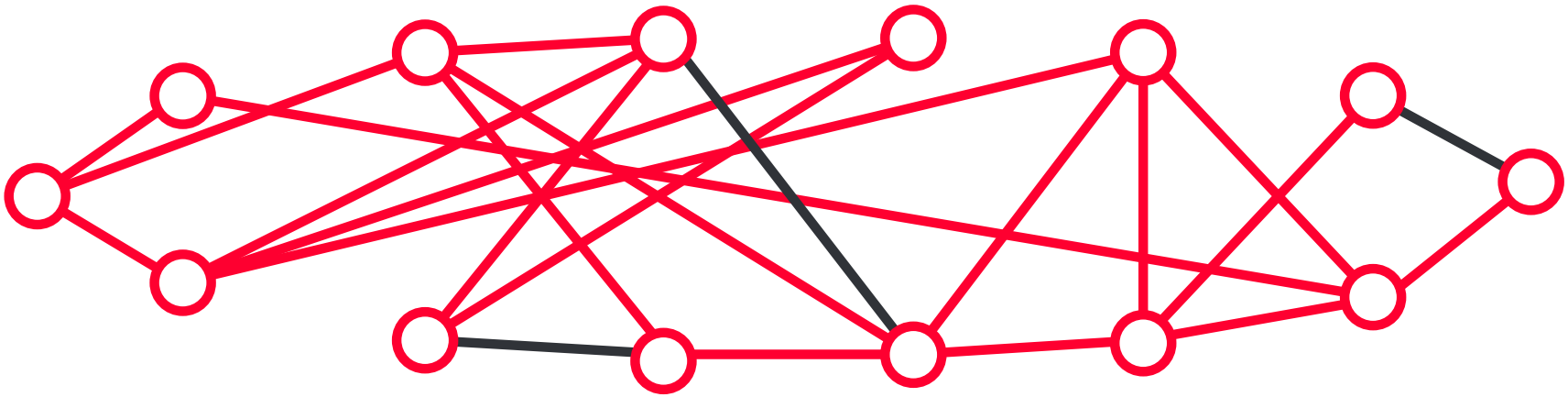
random rewiring



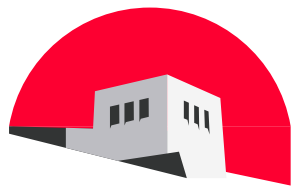
start from the original graph
for ergodicity, edge triples have to be swapped too



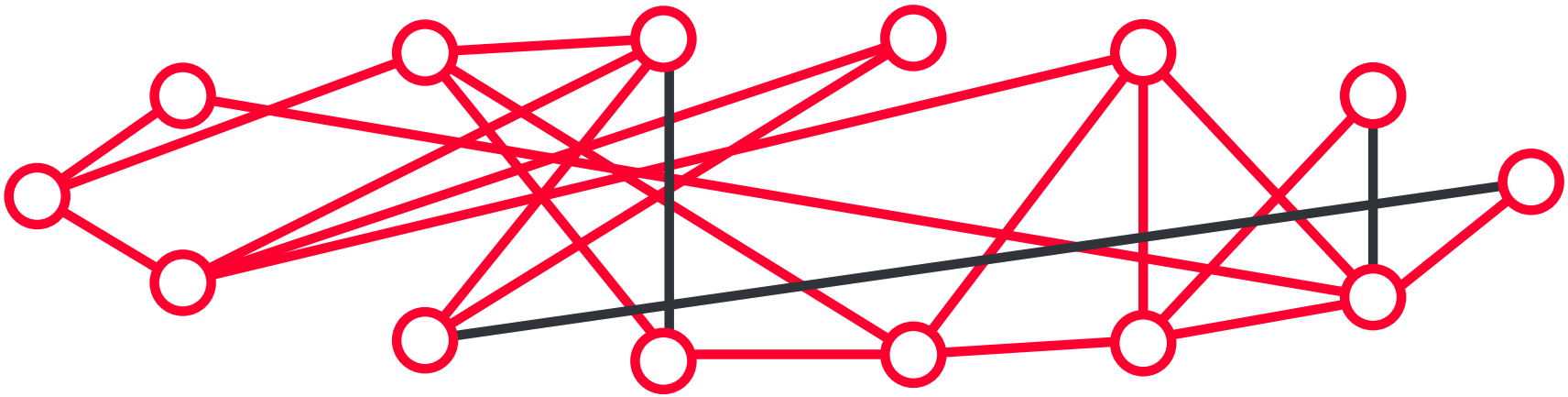
random rewiring



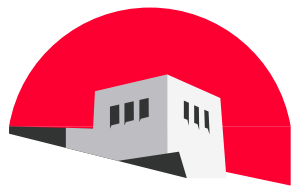
start from the original graph
for ergodicity, edge triples have to be swapped too



random rewiring



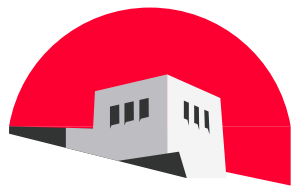
start from the original graph
for ergodicity, edge triples have to be swapped too



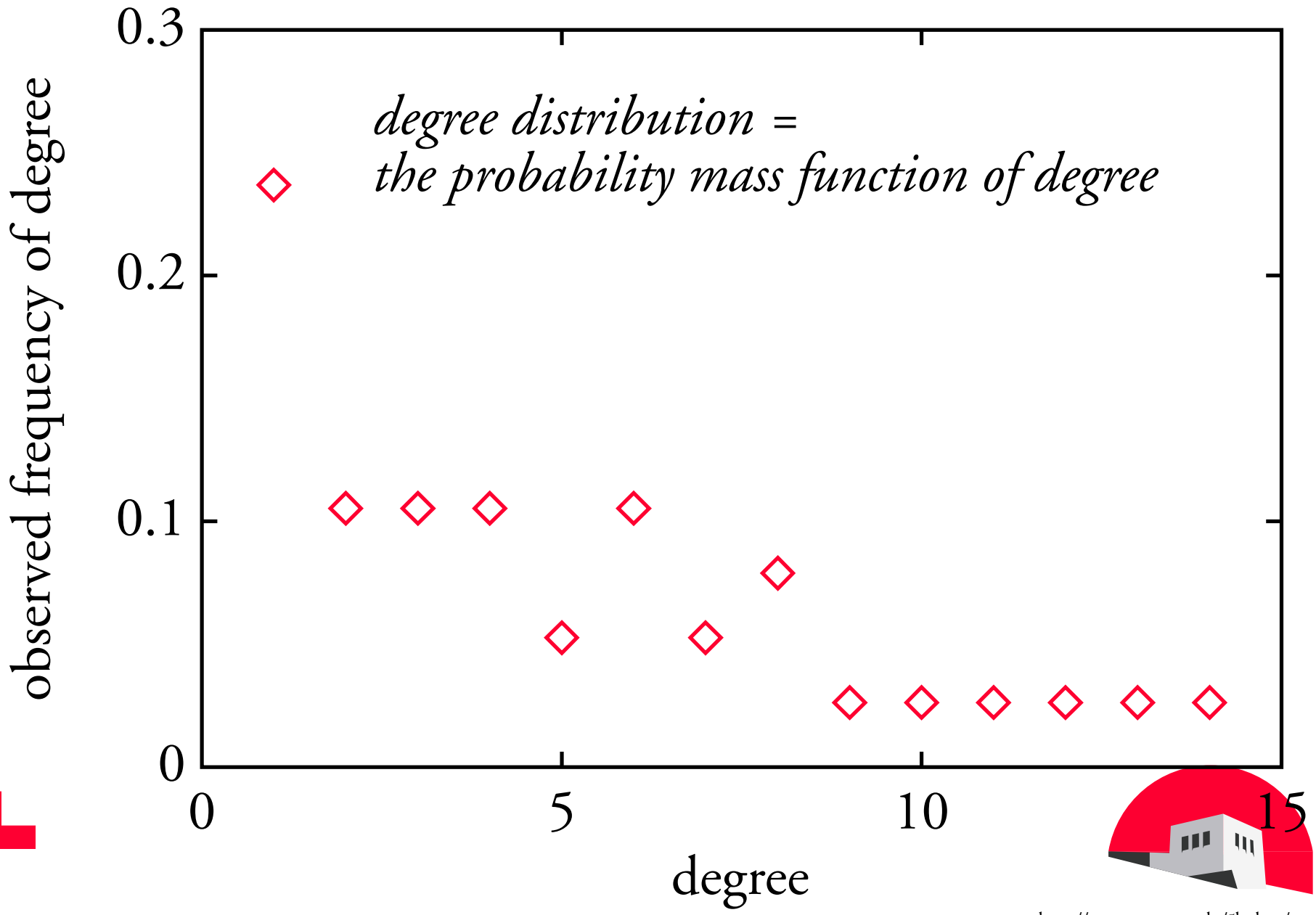
component sizes

giant component 38 vertices

$S = 38$, $s = 38/46 \approx 0.82\cdots$

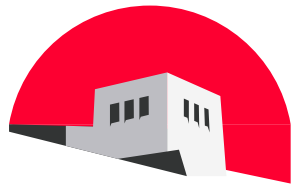


degree distribution



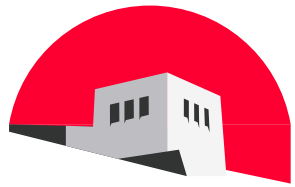
degree distribution

- If the degree distribution looks like a power-law it is called a *scale-free network*.



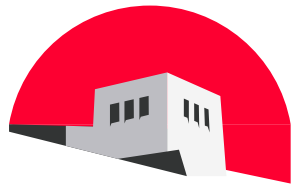
degree distribution

- If the degree distribution looks like a power-law it is called a *scale-free network*.
- This does not necessarily mean system is self-similar. There can be scales visible in other network quantities.

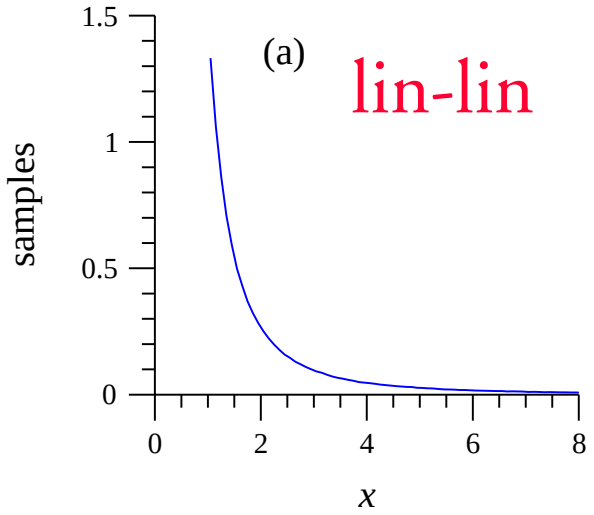


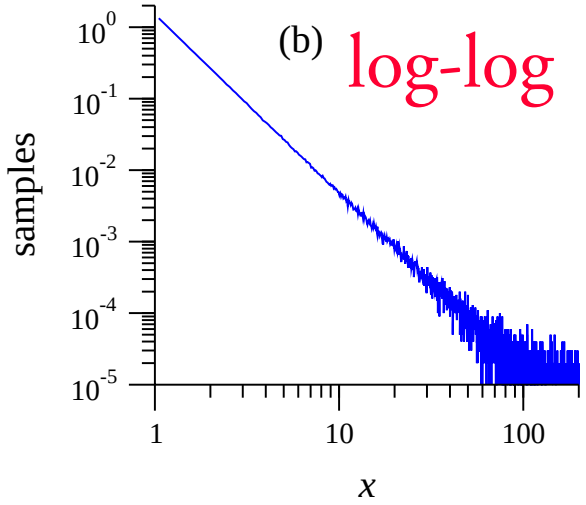
degree distribution

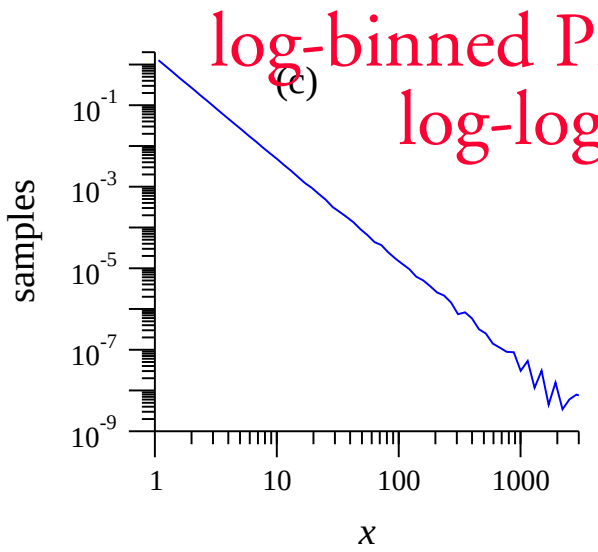
- If the degree distribution looks like a power-law it is called a *scale-free network*.
- This does not necessarily mean system is self-similar. There can be scales visible in other network quantities.
- Exactly where to draw the line between scale-free / not scale-free networks is not known.

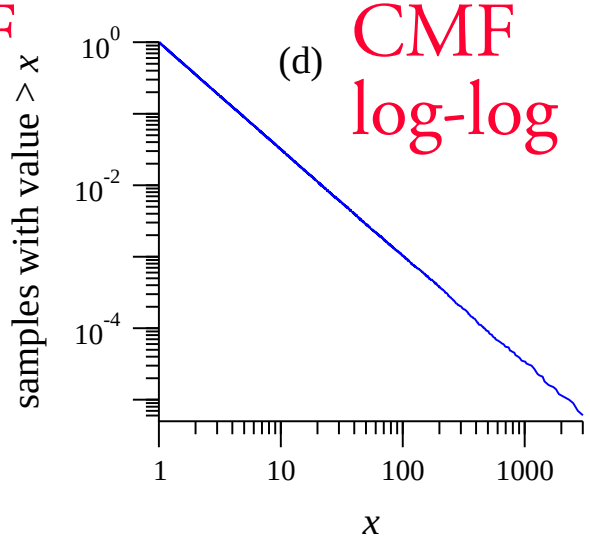


plotting power-laws

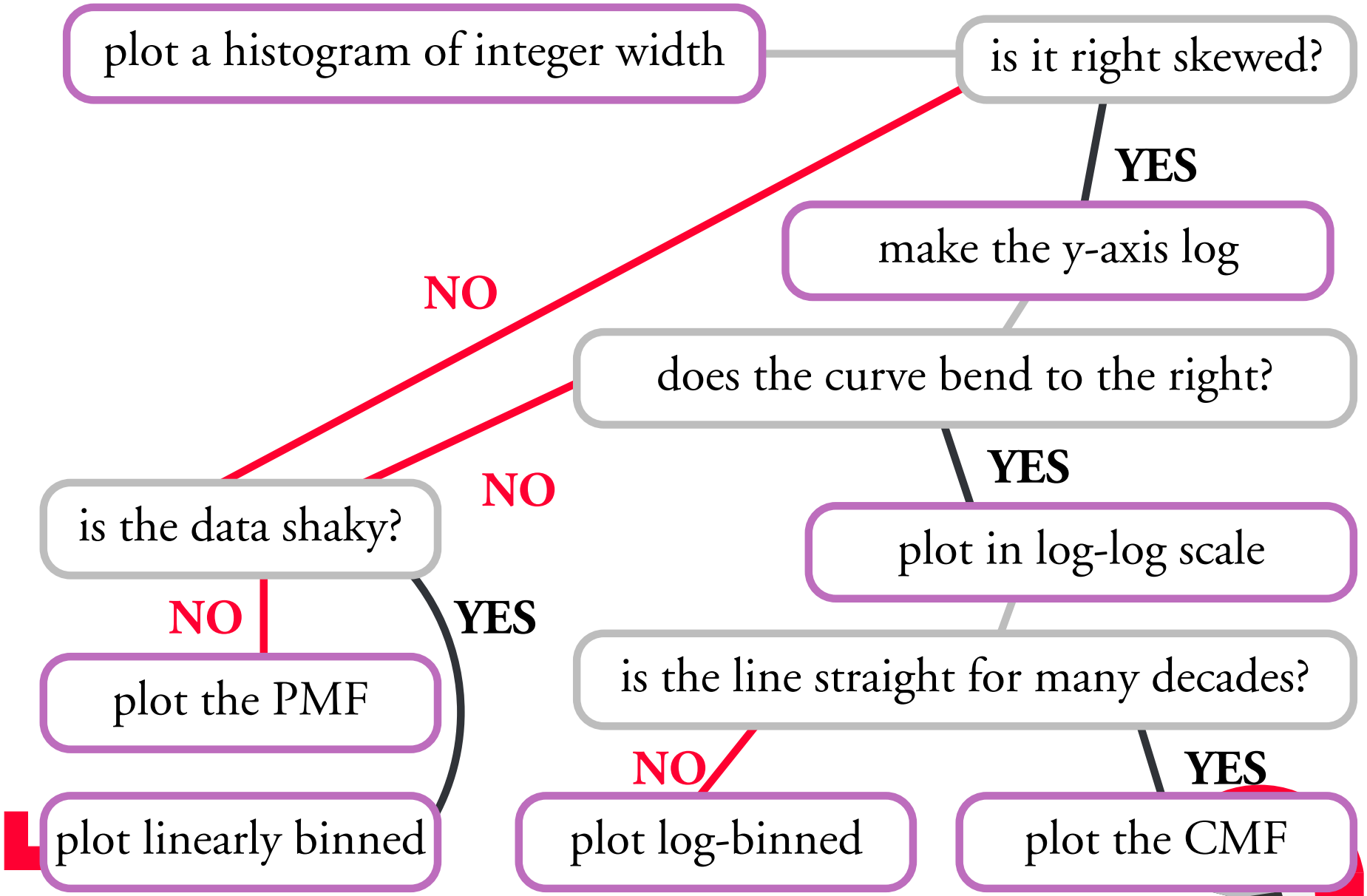






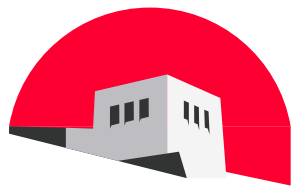


plotting degree distributions



assortative mixing coefficient

Are high-degree vertices connected to other high-degree vertices? Or are they vertices primarily connected to low-degree vertices.



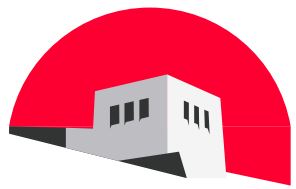
assortative mixing coefficient

Are high-degree vertices connected to other high-degree vertices? Or are they vertices primarily connected to low-degree vertices.

The **assortative mixing coefficient**:

$$r = \frac{4\langle k_1\, k_2\rangle - \langle k_1 + k_2\rangle^2}{2\langle k_1^2 + k_2^2\rangle - \langle k_1 + k_2\rangle^2}$$

where k_i is the degree of the i'th argument of the edges as they appear in an enumeration of the edges.



assortative mixing coefficient

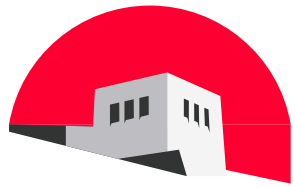
Are high-degree vertices connected to other high-degree vertices? Or are they vertices primarily connected to low-degree vertices.

The **assortative mixing coefficient**:

$$r = \frac{4\langle k_1\, k_2\rangle - \langle k_1 + k_2\rangle^2}{2\langle k_1^2 + k_2^2\rangle - \langle k_1 + k_2\rangle^2}$$

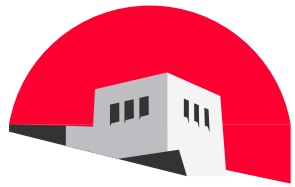
where k_i is the degree of the i'th argument of the edges as they appear in an enumeration of the edges.

In the SSCB name network: $r = 0.023$. In the null-model: $r = -0.23$.



clustering coefficient

How many triangles are there in the network?

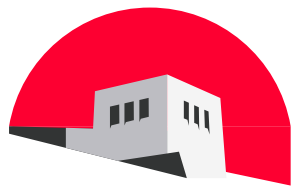


clustering coefficient

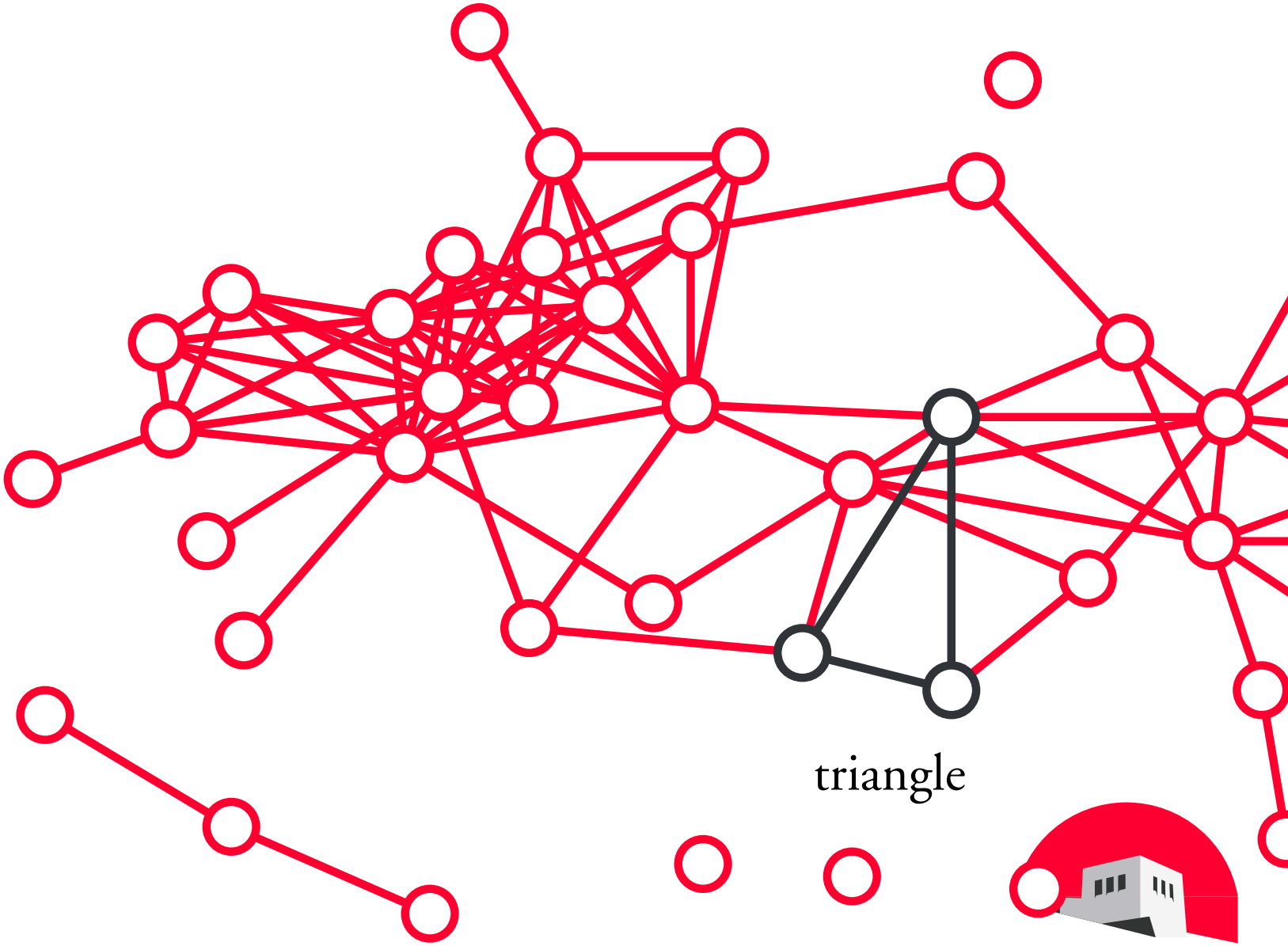
How many triangles are there in the network?

The **clustering coefficient**:

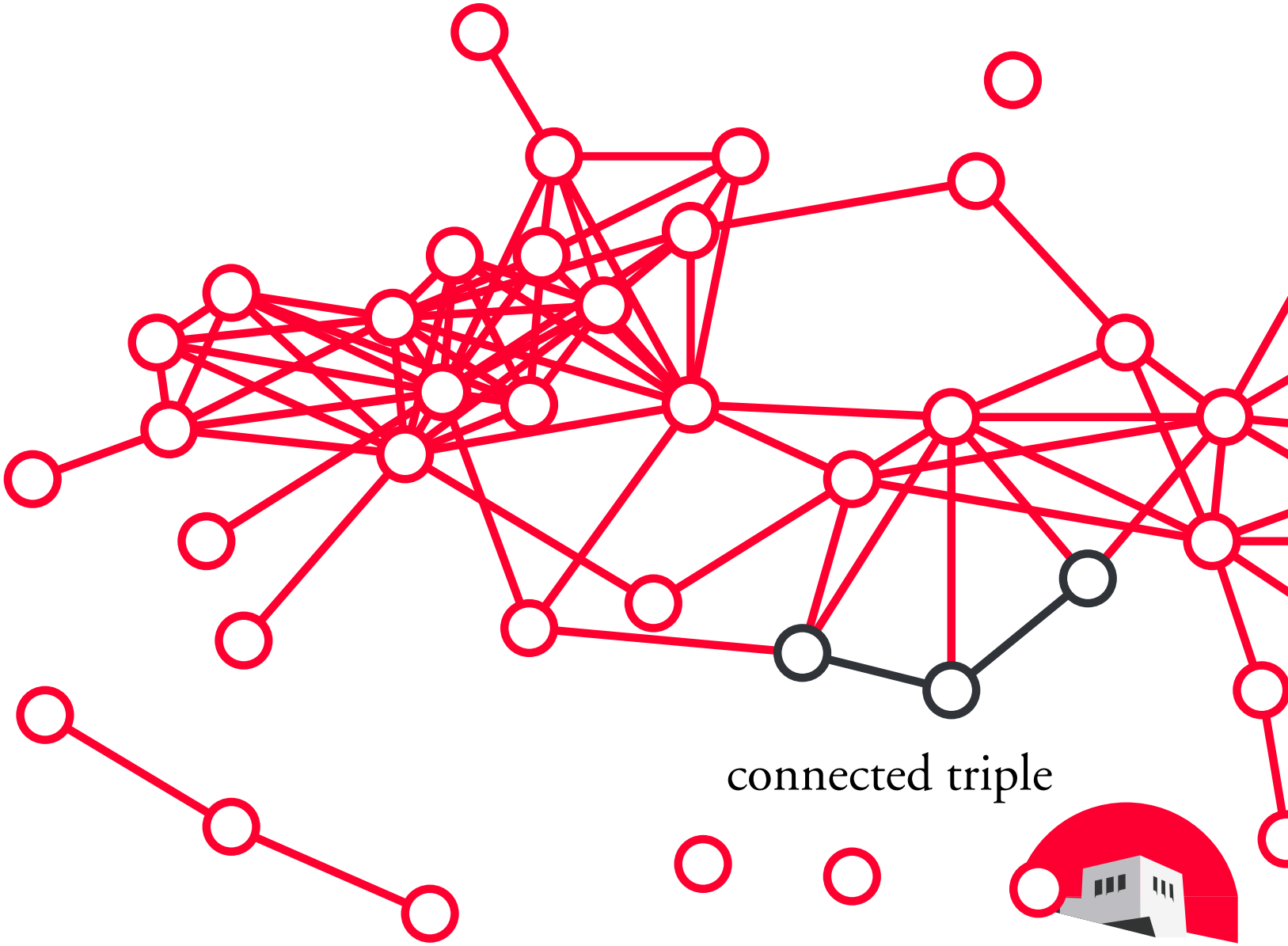
$$C = \frac{\text{the number of triangles}}{3 \times \text{the number of connected triples of vertices}}$$



clustering coefficient



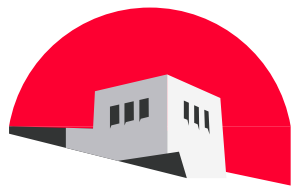
clustering coefficient



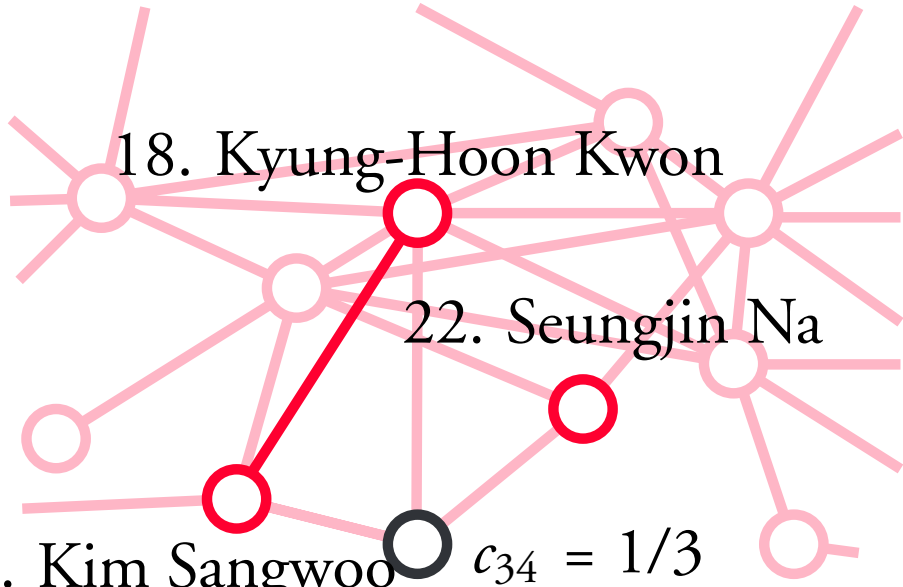
clustering coefficient

Clustering coefficient for the SSCB name network: $C = 0.51$. In the null-model: $C = 0.23$.

The SSCB name network have high clustering and positive assortative mixing coefficient—the same as many social networks.

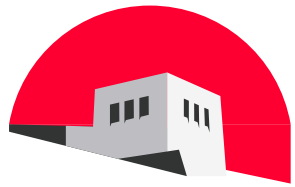


local clustering coefficient



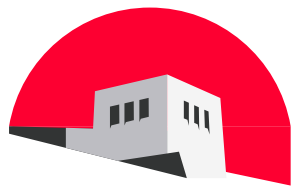
$c_{34} = 1/3$

$$c_i = (\#\text{ edges in neighborhood of } i) / \frac{k_i}{2}$$



centrality measures

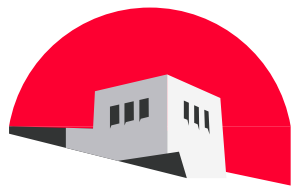
Closeness centrality: $C_C(i)$ The reciprocal average distance from i to all other vertices.



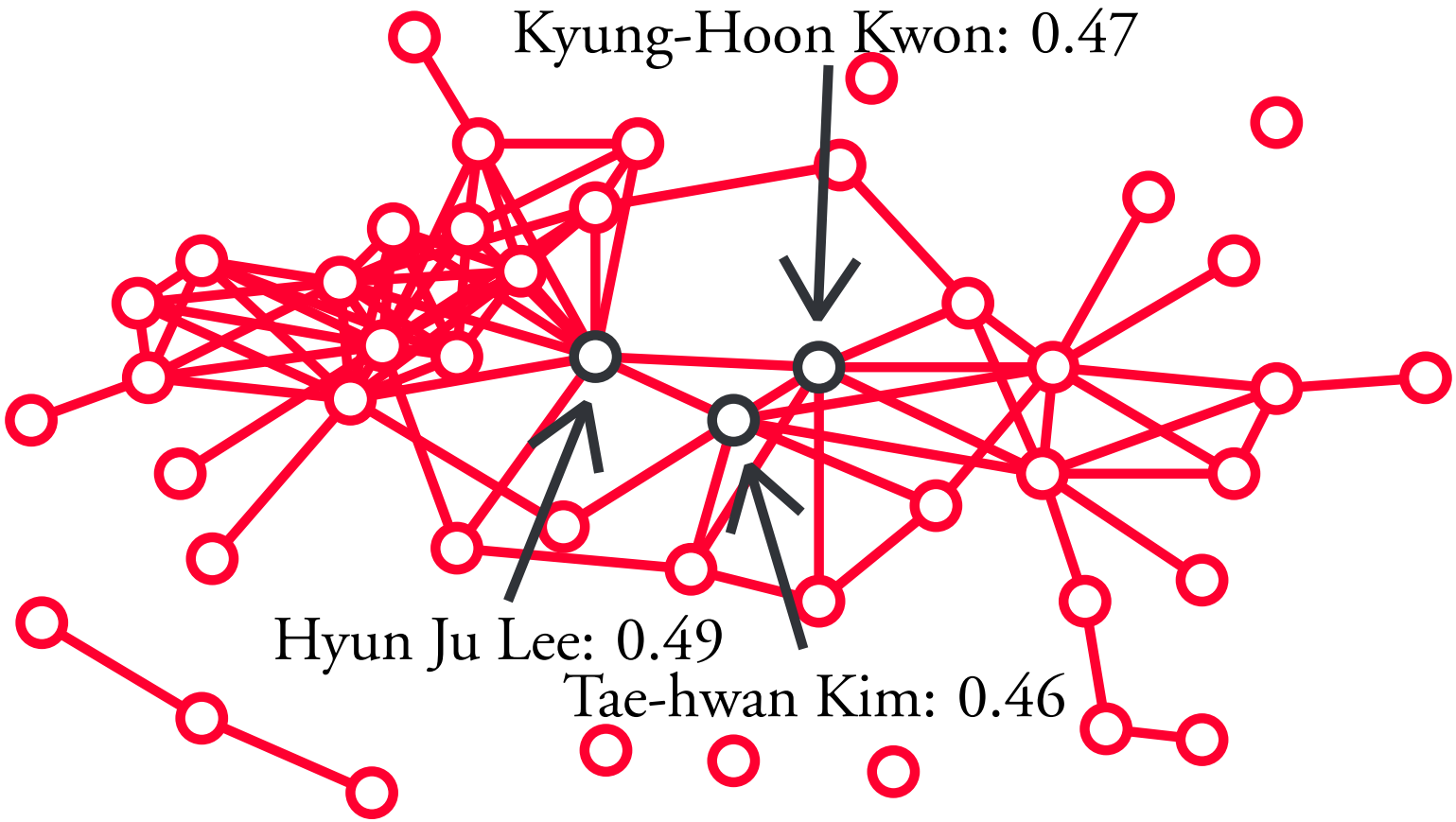
centrality measures

Closeness centrality: $C_C(i)$ The reciprocal average distance from i to all other vertices.

Betweenness centrality: $C_B(i)$ The number of shortest paths between pairs of vertices in the network passing through i.



closeness



betweenness