Brain Cell Analysis (Psy 113L, Winter 2019)

Skirmantas Janusonis (UCSB)

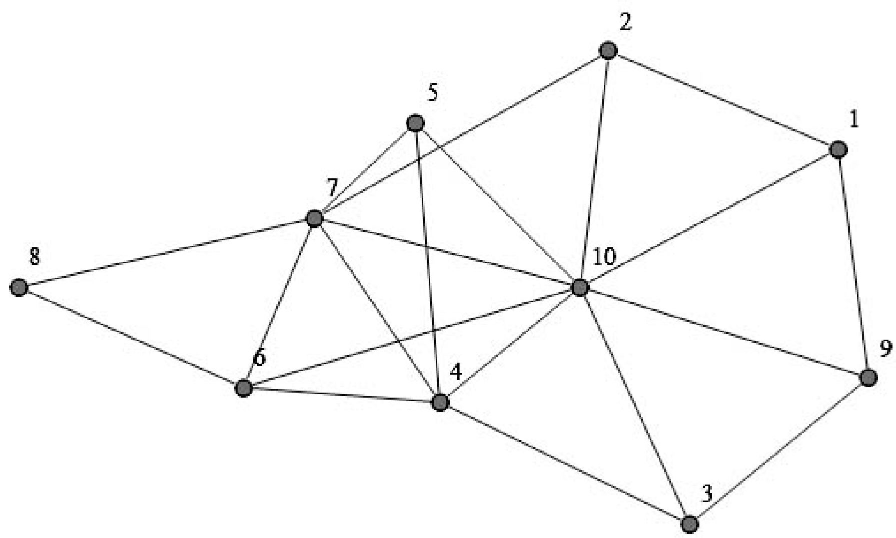
COMPLEX NETWORKS AND ELEMENTS OF GRAPH THEORY

1.1. Graphs and Subgraphs

An early result in experimental network analysis (obtained by Stanley Milgram in the 1960s):

It takes approximately 6 steps in the global social network to connect any two living persons on the planet.

A **graph** is a mathematical object that represents a **network** of nodes connected with links.



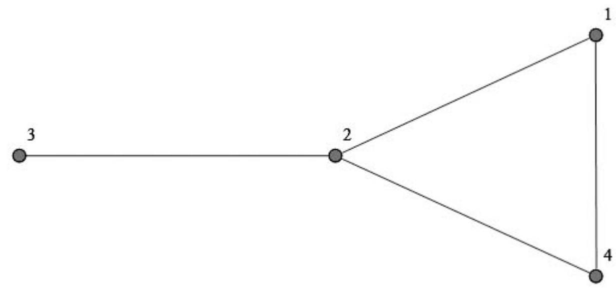
In graph theory:

- GRAPH = network
- VERTEX = node
- EDGE = link

The graph above has 10 vertices and 20 edges.

The **adjacency matrix** of a graph is a *square* matrix in which both the rows and columns represent the vertices. Each element of the matrix can be either 0 (no connection between the two vertices) or 1 (a connection between the two vertices). Below is an example of an adjacency matrix and its corresponding graph:

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

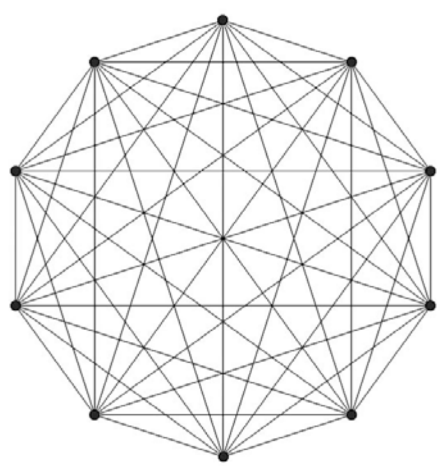


If you know the adjacency matrix of your network (*MyMatrix*), you can create its graph with

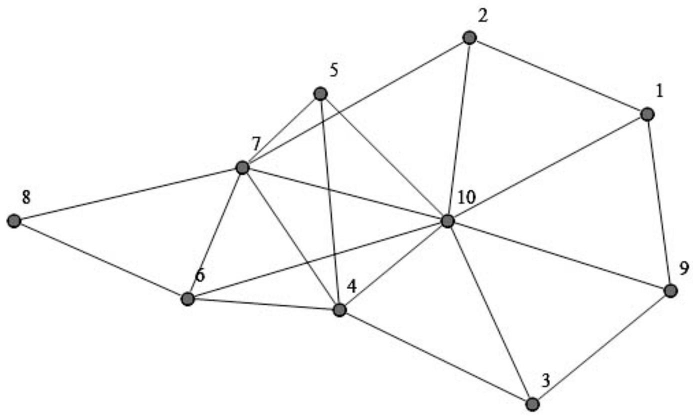
AdjacencyGraph[*MyMatrix*] or
AdjacencyGraph[*MyMatrix*, VertexLabels → "Name", ImagePadding → 10]

A **complete graph** is a graph in which every pair of vertices is connected by an edge.
A complete graph with N vertices has $N(N-1)/2$ edges.

Use CompleteGraph[*N*]



A ***k*-clique** of a graph (with *N* vertices) is a subset of its (*k*) vertices, every pair of which is connected by an edge. In other words, a *k*-clique is a complete graph (with *k* vertices) within a larger (typically, incomplete) graph. Put yet another way, a *k*-clique is a group of friends who know each other (within a larger group of people). In molecular biology, a clique may be formed by a set of functionally related genes that interact with each other.



For example, vertices 5, 7, and 10 form a *3*-clique in the graph above.

Do you see more 3-cliques?
Do you see 4-cliques?
Find the number of student cliques in the class.

A **maximal clique** is a clique that can't be extended by including one more adjacent vertex (i.e., a clique that does not exist within the vertex set of a larger clique).

Is the {5, 7, 10} clique maximal in the graph above?
Find a maximal clique in the class.

A **maximum clique** is a clique of the largest possible size in a given graph. The **clique number** of a graph is the number of vertices in a maximum clique.

Is the {5, 7, 10} clique maximum in the graph above?
Find the maximum clique(s) in the class and the class clique number.

Use FindClique[*MyGraph*]

The **community** has several definitions in graph theory. Typically, it is a set of vertices that "almost" form a clique, but are a little more loosely connected. Some authors define the **community** as the **union of all adjacent *k*-cliques**. Two cliques are *adjacent* if they share all but one vertex (*k – 1*).

Consider the following example. For example, James, Mary, and John know each other (form a *3*-clique). Also, Mary, John and Jennifer know each other (form another *3*-clique). These two cliques are *adjacent* because Mary and John are shared by both cliques. Therefore, James, Mary, John, and Jennifer form a *community*. They don't form a *4*-clique, since James doesn't know Jennifer.

Analysis of *k*-cliques and their communities is a hard ("NP-complete") computational problem, especially for large networks. Consider using **CFinder**, a free and intuitive program developed by well-known graph theoreticians (http://www.cfinder.org/). It has been used to analyze large genetic and social networks.

A ***k*-cycle** is a set of *k* different vertices that form a closed loop. Another way to put it, a cycle is a closed path: by following it, you eventually end up in the same place.

A set of *k* **independent vertices** has no edges connecting them. You can think of them as people who don't know each other, or the opposite of the clique. In the graph above, vertices 2, 4, 8 and 9 are independent.

You can find the largest independent set of a graph with

FindIndependentVertexSet[*MyGraph*]

The **Ramsey number** *R(r, s)* is the smallest number of vertices that guarantees that the graph will contain **either** *r* completely connected vertices (i.e., an *r*-clique) **or** *s* independent vertices.

Other (more intuitive but equivalent) formulations:

Dinner Party Problem: *How many people do you have to invite over for dinner to make sure that either at least* ***r*** *people know each other (form an r-clique)* <u>*or*</u> *at least* ***s*** *people don't know each other?*

Neuroanatomy Problem: *How many brain structures do you have to consider to guarantee that at least* ***r*** *structures are completely interconnected* <u>*or*</u> *at least* ***s*** *structures are not directly connected?*

Ramsey Numbers

r,s	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	1	2	3	4	5	6	7	8	9	10
3	1	3	6	9	14	18	23	28	36	40–43
4	1	4	9	18	25	35–41	49–61	56–84	73–115	92–149
5	1	5	14	25	43–49	58–87	80–143	101–216	125–316	143–442
6	1	6	18	35–41	58–87	102–165	113–298	127–495	169–780	179–1171
7	1	7	23	49–61	80–143	113–298	205–540	216–1031	233–1713	289–2826
8	1	8	28	56–84	101–216	127–495	216–1031	282–1870	317–3583	317-6090
9	1	9	36	73–115	125–316	169–780	233–1713	317–3583	565–6588	580–12677
10	1	10	40–43	92–149	143–442	179–1171	289–2826	317-6090	580–12677	798–23556

(A range is given when the precise number is not known.)

It is known that, **if *r* = *s* ≥ 3**,

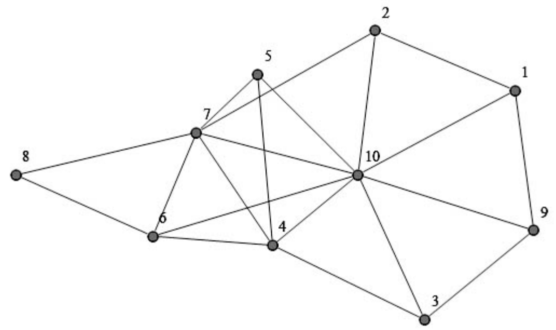
$$R(r,s) > 2^{r/2}$$

[Diestel (2006), p. 296]

Ramsey numbers can be calculated in Mathematica.
See http://mathworld.wolfram.com/notebooks/GraphTheory/RamseyNumber.nb .

1.2. Some Terminology

Two vertices are **neighbors** if they are connected by an edge.



Use NeighborhoodGraph[*MyGraph*, *VertexID*]

In the graph above, are vertices 6 and 10 neighbors? How about 1 and 3?
By using the neighbors *term, propose a quick definition of the complete graph.*

The **degree (*k*) of a vertex** is the number of edges connected to this vertex (i.e., the number of the vertex's neighbors).

Use VertexDegree[*MyGraph*]

If you are a vertex in the graph of the global social network, what is your vertex-degree?

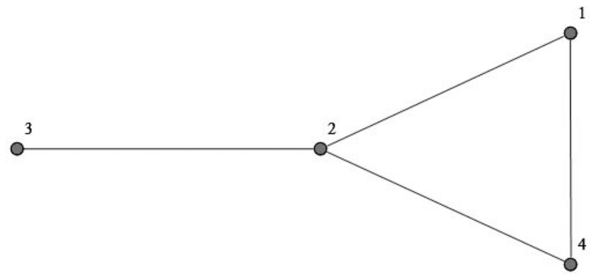
The **coordination number** *z* is the *mean* number of edges per vertex (i.e., the *mean* number of neighbors per vertex, or the mean vertex degree).

$$z = \frac{2\ EdgeNumber}{VertexNumber}$$

Use:
2 EdgeCount[*MyGraph*]/VertexCount[*MyGraph*] or, equivalently,
Mean[VertexDegree[*MyGraph*]]

In your opinion, is your vertex-degree larger or smaller than the coordination number of the global social network?

The **eccentricity of a vertex** is the largest degree of separation between *this vertex* and each of the other vertices in the graph. In other words, it is the longest of the shortest paths from the vertex to each of the other vertices.



In the graph above, the eccentricity of vertex #1 is 2 (the number of steps to get to the farthest vertex (#3)) and the eccentricity of vertex #2 is 1 (only one step is needed to get to the other vertices).

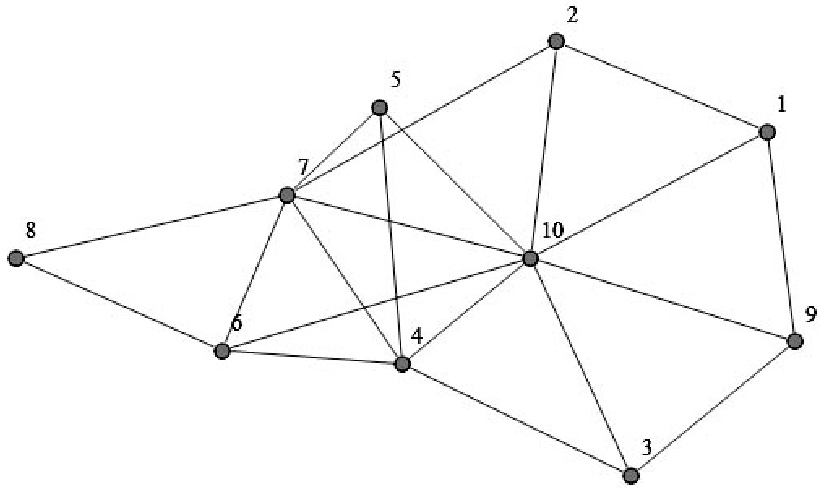
Use VertexEccentricity[*MyGraph*, *VertexID*]

The **graph diameter (*D*)** is the largest degree of separation between *all pairs of vertices* in the graph. In other words, it is the maximal eccentricity present in the graph. The graph above has the diameter of 2.

Use GraphDiameter[*MyGraph*]

What is the graph diameter in a complete graph?
What is the approximate graph diameter in the global community of people?

The **clustering coefficient (*C*)** is the mean fraction of the pairs of neighbors of a vertex that are also neighbors of each other.



Manually determine the clustering coefficient in the graph above. Remember that N elements can make N(N−1)/2 distinct pairs. Therefore, 4 neighbors will make 6 pairs, each of which has to be checked for direct connectivity.

Use MeanClusteringCoefficient[*MyGraph*]//N (Mathematica 9)

Mathematica 8 has no function to calculate the clustering of a graph. Here is my code that does the calculation:

```
VertexN = VertexCount[MyGraph];
ConnectedProportion = Table[-1, {VertexN}];

For[i = 1, i ≤ VertexN,

        NeighborPairs =
UndirectedEdge@@@Subsets[DeleteCases[VertexComponent[MyGraph, {i}, 1], i], {2}];
        If[Length[NeighborPairs] ≠ 0,
        ConnectedProportion[[i]] = N[Count[EdgeQ[MyGraph, #]&/@NeighborPairs,
True]/Length[NeighborPairs]]];

        i++];

Mean[DeleteCases[ConnectedProportion, -1]]
```

1.3. Erdös-Rényi Graphs

The **Erdős-Rényi graph** is a random graph in which every pair of *N* vertices is connected (by an edge) with the same probability *p*.

If the graph is known to be Erdős-Rényi but the value of *p* is unknown, it can be roughly estimated by taking the number of the actual edges and dividing it by the number of edges in a *complete* graph with the *same number of vertices*. This number is *N(N−1)/2*. In coin tossing, the equivalent procedure would be counting the number of "heads" and dividing this number by the maximal possible number of "heads" (i.e., the total number of tosses).

Use:
RandomGraph[{*VertexNumber*, *EdgeNumber*}] or
RandomGraph[BernoulliGraphDistribution[*VertexNumber*, *p*]]

The following lists some **properties of Erdős-Rényi graphs** (*N*, *p*).
Some, but not all, of these properties can be extended to *all* random graphs (not just the ones of the Erdős-Rényi type).

You will need this function.
Definition: $C_n^m = \frac{n!}{m!(n-m)!}$, where $n! = 1 * 2 * 3 \dots * n$. By definition, $0! = 1$.
Explanation: It is the number of different ways one can select *m* objects from *n* objects.
Example: $C_3^2 = \frac{3!}{2!(3-2)!} = \frac{6}{2*1} = 3$.
Calculate $C_n^m$by using Binomial[n, m]

The probability that a **vertex** will be of ***k*-degree** (i.e., will have *k* neighbors or *k* edges directly connected to it) is

$$p_k = C_{N-1}^k p^k (1-p)^{N-1-k},$$

and, when *N* is large, can be approximated by

$$p_k = \frac{(pN)^k}{k!} e^{-pN}$$

The **coordination number** (mean vertex degree) is

$$z = p(N-1) \approx pN.$$

The **graph diameter** D is proportional to

$$\log(N) / \log(z).$$

The clustering coefficient:

$$C = \frac{z}{N-1} \approx \frac{z}{N} = p.$$

Table 1 Empirical examples of small-world networks

	L_{actual}	L_{random}	C_{actual}	C_{random}
Film actors	3.65	2.99	0.79	0.00027
Power grid	18.7	12.4	0.080	0.005
C. elegans	2.65	2.25	0.28	0.05

Characteristic path length L and clustering coefficient C for three real networks, compared to random graphs with the same number of vertices (n) and average number of edges per vertex (k). (Actors: $n = 225{,}226$, $k = 61$. Power grid: $n = 4{,}941$, $k = 2.67$. *C. elegans*: $n = 282$, $k = 14$.) The graphs are defined as follows. Two actors are joined by an edge if they have acted in a film together. We restrict attention to the giant connected component[16] of this graph, which includes ~90% of all actors listed in the Internet Movie Database (available at http://us.imdb.com), as of April 1997. For the power grid, vertices represent generators, transformers and substations, and edges represent high-voltage transmission lines between them. For *C. elegans*, an edge joins two neurons if they are connected by either a synapse or a gap junction. We treat all edges as undirected and unweighted, and all vertices as identical, recognizing that these are crude approximations. All three networks show the small-world phenomenon: $L \gtrsim L_{\text{random}}$ but $C \gg C_{\text{random}}$.

[Watts & Strogatz (1998)]

The probability that a graph contains a *k*-clique is

$$P \leq C_N^k p^{C_k^2}$$

[Diestel (2006), p. 296]

The probability that a graph contains a set of *k* independent vertices is, analogously,

$$P \leq C_N^k q^{C_k^2},$$

where $q = 1 - p$.

[Diestel (2006), p. 296]

The expected number of *k*-cliques:

$$C_N^k p^{\frac{k(k-1)}{2}} (1 - p^k)^{N-k}$$

[Gros (2008), p. 6]

If *p* = 0.5, the clique number (the size of the maximum clique) is almost precisely

$$2 \log_2 N$$

[Wikipedia.org]

The expected number of *k*-cycles is

$$\frac{N(N-1)(N-2) \dots (N-k+1)}{2k} p^k$$

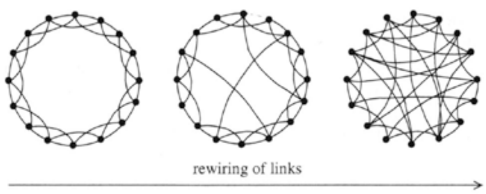
[Diestel (2006), p. 298]

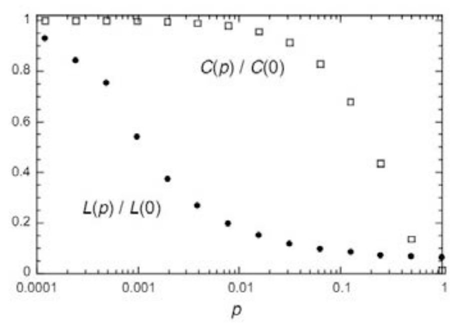
1.4. Small-World Graphs

Erdős-Rényi graphs may appear complex, but they are relatively easy in the sense that they are completely random. **Typically, mathematics has many good tools to deal with systems that are either completely *deterministic* or completely *random*.**

Most real complex systems (such as brains and societies) **are neither completely deterministic nor completely random**. For example, living systems are often said to exist “at the edge of chaos”: they appear incredibly well-organized, but can be surprisingly random at the molecular and population levels. To deal with these systems, we need to better understand relationships between determinism and randomness and may be forced to develop completely new mathematical approaches. Modern mathematics has discovered striking examples of deterministic systems producing random behavior and random systems producing deterministic behavior.

Watts-Strogatz Model





Characteristic path length $L(p)$ and clustering coefficient $C(p)$ for the family of randomly rewired graphs described in Fig. 1. Here L is defined as the number of edges in the shortest path between two vertices, averaged over all pairs of vertices. The clustering coefficient $C(p)$ is defined as follows. Suppose that a vertex v has k_v neighbours; then at most $k_v(k_v - 1)/2$ edges can exist between them (this occurs when every neighbour of v is connected to every other neighbour of v). Let C_v denote the fraction of these allowable edges that actually exist. Define C as the average of C_v over all v. For friendship networks, these statistics have intuitive meanings: L is the average number of friendships in the shortest chain connecting two people; C_v reflects the extent to which friends of v are also friends of each other; and thus C measures the cliquishness of a typical friendship circle. The data shown in the figure are averages over 20 random realizations of the rewiring process described in Fig. 1, and have been normalized by the values $L(0)$, $C(0)$ for a regular lattice. All the graphs have $n = 1{,}000$ vertices and an average degree of $k = 10$ edges per vertex. We note that a logarithmic horizontal scale has been used to resolve the rapid drop in $L(p)$, corresponding to the onset of the small-world phenomenon. During this drop, $C(p)$ remains almost constant at its value for the regular lattice, indicating that the transition to a small world is almost undetectable at the local level.

[Watts & Strogatz (1998)]

Use:

RandomGraph[WattsStrogatzGraphDistribution[*VertexNumber*,*RewiringProbability*]] or
RandomGraph[WattsStrogatzGraphDistribution[*VertexNumber*,*RewiringProbability*],
GraphLayout → “CircularEmbedding”]

Barabási-Albert Model

In the Barabási-Albert model, “popular” vertices tend to get even more edges (i.e., can go “viral” in the network). In this model, the probability that a **vertex** will be of ***k*-degree** (i.e., will have *k* neighbors or *k* edges directly connected to it) is

$$p_k \sim k^{-\alpha}, \text{ where } \alpha = 3.$$

Use:

RandomGraph[BarabasiAlbertGraphDistribution[*n*, *k*]]

1.5. Distribution of Vertex Degrees

In Mathematica, the **distribution of vertex degrees** in a graph (MyGraph) can be plotted with the following commands (→ should be typed as '-' and '>' with no space between them):

```
VertexDegrees=Tally[VertexDegree[MyGraph]];
ListPlot[VertexDegrees,PlotRange→{{0,Automatic},{0,Automatic}},Filling→Axis,
FillingStyle→Thin]
```

If the distribution follows the power law, the **power α can be estimated as follows** (assuming the adjacency matrix is symmetric (i.e., the graph is undirected) and the previous two commands have already been run):

```
(* The following command drops unconnected vertices to avoid Log[0]. It can be skipped if these
vertices have already been dropped. In this case, make sure to set
VertexDegreesCorrected=VertexDegrees. *)

VertexDegreesCorrected=Drop[VertexDegrees,Flatten[Position[VertexDegrees[[All,1]],0]]];

(* The following commands drop hyper-connected vertices above a vertex degree threshold
(MaxCutOff). In some cases, they may distort the distribution. Always see the plot first. These
commands can be skipped. *)

MaxCutOff=10; (* Or any other degree threshold. *)
VertexDegreesCorrected =  Delete[VertexDegreesCorrected,
Flatten[Position[VertexDegreesCorrected[[All, 1]], #] & /@Select[VertexDegreesCorrected[[All,
1]], # > MaxCutOff &], 1]];

ListPlot[VertexDegreesCorrected,PlotRange→{{0,Automatic},{0,Automatic}},Filling→Axis,
FillingStyle→Thin]

(* The actual calculation of α. *)

VertexDegreesLog=Log[N[VertexDegreesCorrected]];
MaxVertexLog=Ceiling[Max[VertexDegreesLog[[All,1]]]];
MyEstimate=LinearModelFit[VertexDegreesLog,x,x]

Show[ListPlot[VertexDegreesLog,PlotRange→{0,Automatic}],  Plot[MyEstimate["BestFit"], {x, 0,
MaxVertexLog}]]

Print["alpha = ",−MyEstimate["BestFit"][[2,1]]]
```

1.6. Percolation

Percolation is a very important property of real networks. Technically, a developing graph undergoes **percolation transition** if one observes the emergence of a **giant connected component (GCC)** that spreads through the entire network. This may not be terribly illuminating. Luckily, percolation can be thought of as “leakiness” and can be easily explained with intuitive examples.

<u>Example 1</u>. A secret is leaked from one node of a network. If the node connectivity is low, the secret spreads to a few neighboring vertices, but can’t travel any further (it soon runs into blind alleys). As the connectivity increases, the secret can now trickle through the entire network. It doesn’t have to reach every single vertex, but it travels through the entire “width” of the network. (If this is not clear, think of a “viral” video. Has it been watched by every single individual? Of course not, but it’s intuitively clear it has spread through the entire network.) If the network grows larger, so does the number of vertices that know the secret. In other words, the network has undergone percolation transition and now contains a GCC.

<u>Example 2.</u> Consider a thick slab of a material that can develop little cracks (think of them as graph edges). Now pour water on the top surface and observe the bottom surface. If the frequency of cracks is low, the slab is unlikely to leak. Some cracks will join to form larger cracks, but there will be few or no uninterrupted channels going all the way from the top to the bottom of the slab. As the number of cracks grows, the slab will become leaky. Technically, it undergoes percolation transition.

<u>Examples of “Good” Percolation:</u>

1. The Internet. If the connectivity of network servers falls below the percolation threshold, the Internet disintegrates into local disconnected networks.
2. The brain. Every part of the brain should be able to reach (directly or directly) any other part of the brain.

<u>Examples of “Bad” Percolation:</u>

1. Leakage of state secrets. The existence of Wikileaks is a good example of an unintended percolation in a network.
2. A leaky roof. There may be no actual hole in the roof but water can still drip because of a GCC formed by microscopic cracks.

Percolation Transition is Rapid and Switch-like

Typically, *percolation transition* in a network happens *suddenly* when a certain parameter (e.g., the p in Erdős-Rényi graphs) reaches a *critical value*. In other words, a network abruptly switches from a very low probability of percolation to a very high probability of percolation.

Percolation Transition in Erdős-Rényi graphs

In the Erdős-Rényi graph with N vertices, percolation transition occurs when the edge connection probability p reaches the critical value of

$$p_c = \frac{1}{N}.$$

Advanced Percolation Research

Percolation is a "hot" and active area of research. Its results are important for neurosciences, social sciences, economics, and security. It has been shown (Palla et al., 2007) that if one is interested in percolation formed not by individual vertices but by *adjacent k*-cliques, the Erdős-Rényi graph will undergo percolation transition at

$$p_c = \frac{1}{[N(k-1)]^{\frac{1}{k-1}}}.$$

Since two vertices connected by an edge can be thought of as a *2*-clique, setting $k = 2$ again yields

$$p_c = \frac{1}{N}.$$

Therefore, the Palla and colleagues' result generalizes simple percolation.

1.7. Robustness of Networks

Robust networks remain above the percolation limit even when many nodes are removed. Most natural networks are robust.

Generally, a network is robust if the probability that a **vertex** will be of ***k*-degree** (i.e., will have *k* neighbors or *k* edges directly connected to it) is

$$p_k \sim k^{-\alpha}, \text{ where } \alpha \approx 2.$$

Such networks are knows as scale-free networks.

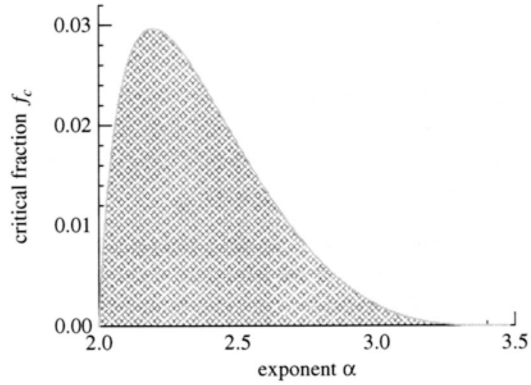
Random Failure of Vertices

Denote z_1 the mean number of *first-degree neighbors* (directly connected to the node) and z_2 the mean number of *second degree neighbors* (connected to the node by two links). Next, denote b_c the *critical fraction of active nodes below which the network loses its GCC* (i.e., falls below the percolation limit). In practice, it is equivalent to network failure.

Then, if $N \to \infty$, three important regimes can be identified (Gros, 2008):

	z_1	z_2	$b_c = z_1 / z_2$	**Robusteness**
$1 < \alpha \leq 2$	$\to \infty$	$\to \infty$	unpredictable	poor
$2 < \alpha \leq 3$	$< \infty$	$\to \infty$	$\to 0$	very good
$3 < \alpha < 3.478$	$< \infty$	$< \infty$	constant	normal
$3.478 < \alpha$	No GCC possible			

Biased Failure of Highest-Degree Vertices



The critical fraction f_c of vertices, Eq. (1.59). Removing a fraction greater than f_c of highest degree vertices from a scale-free network, with a power-law degree distribution $p_k \sim k^{-\alpha}$ drives the network below the percolation limit. For a smaller loss of highest degree vertices (*shaded area*) the giant connected component remains intact (from Newman, 2002)

One can see in the graph that:

- Removal of more than ~3% of the highest-degree vertices always leads to destruction of the GCC. The maximal robustness is achieved for $\boldsymbol{\alpha \approx 2.2}$, which is close to what has been observed in real-world networks.
- Networks with $\alpha > 3.4788$ can have no GCC.

Cited Literature

Diestel R. (2006) *Graph Theory* (Third Ed., Springer)

Gros C. (2008) *Complex and Adaptive Dynamical Systems: A Primer* (Springer).

Newman M.E.J. (2002) Random Graphs and Models of Networks. http://arxiv.org/abs/cond-mat/0202208 .

Palla G., Derenyi I., Vicsek T. (2007) The critical point of *k*-clique percolation in the Erdős-Rényi graph. *J. Stat. Physics* 128: 219-227.

Watts, D.J. Strogatz, S.H. (1998) Collective dynamics of small world networks. *Nature* 393: 440-442.