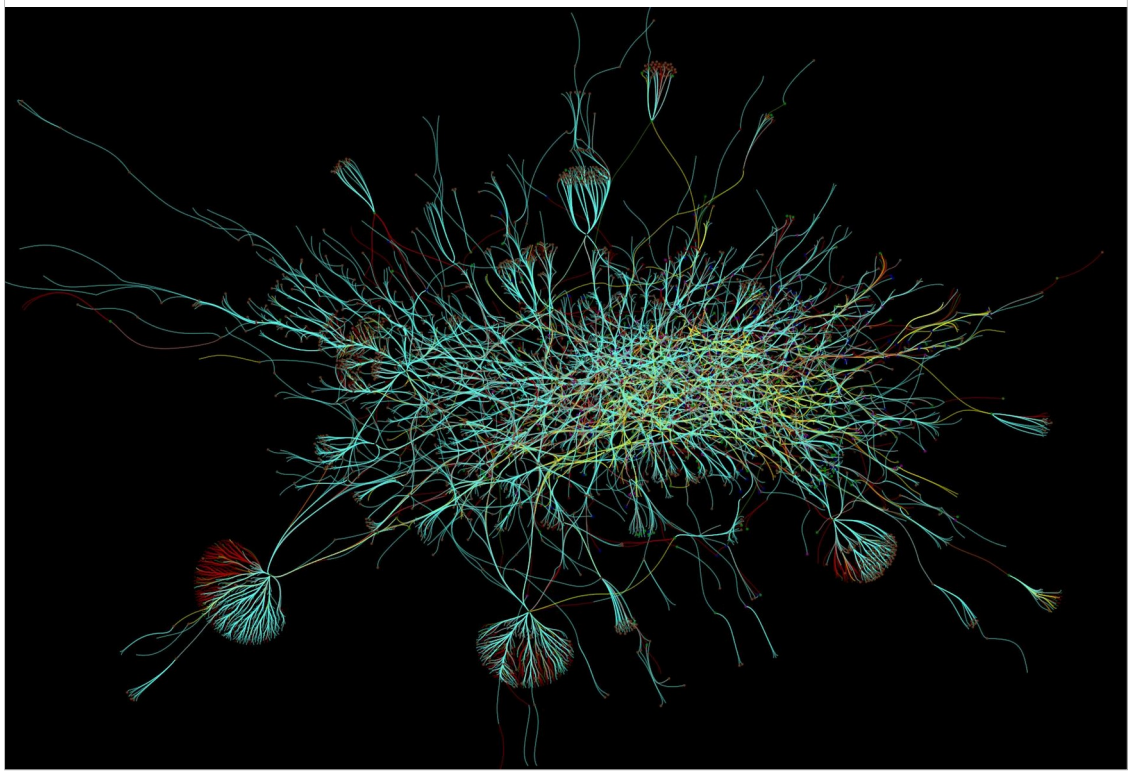
Networks



Online materials

- Barabási's book (also in print, also in hungarian):
  - http://barabasi.com/networksciencebook/
- Online courses:
  - https://github.com/ladamalina/coursera-sna
  - https://www.coursera.org/learn/python-social-network-analysis
- Wikipedia

This class will only be a quick intro to network science and network visualization. For more in-depth exploration, I can recommend these online materials.

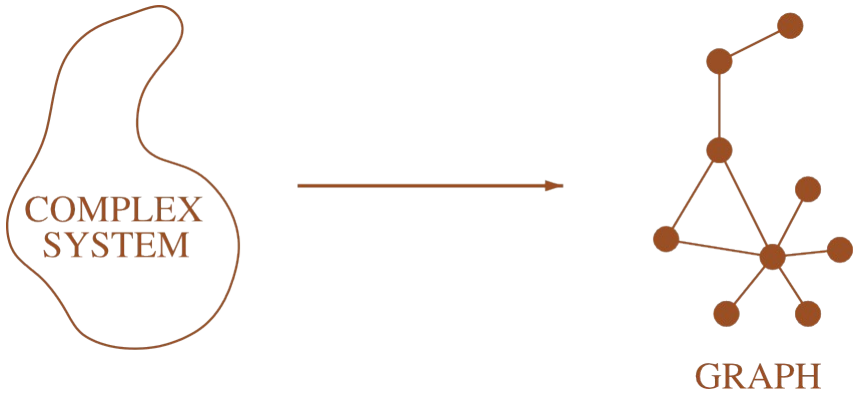
Graph theory vs. network science vs. systems science vs. ….

- Interdisciplinary area
- Difference is in approach & focus

- Math-centric viewpoint: graph theory
  - existence theorems (eg. Szemerédy regularity lemma)
  - worst-case / pathological cases
- Physicist viewpoint: network science
  - Mean field approximations
  - “typical / average cases”

As an interdisciplinary area, most people working on networks actually have a background in physics, math, biology, sociology or other area. The terminology and methods they use will be strongly influenced by this background.

Why networks?



The fundamental idea in network science is that we use a graph to model the system being studied. Surprisingly, the graphs describing many different systems have similar statistical properties – this universality allows using the same mathematical methods on different systems.

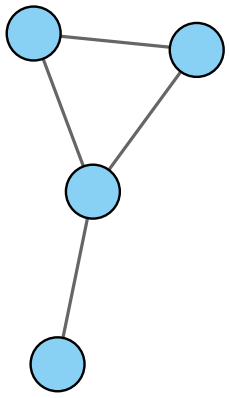
Important to note that the graph being studied is not the original system, just an approximation / abstraction of it.

Defining a network

When using a model to study a given system, deciding the level of detail we want to use is very important: adding more detail to the model will make it more accurate, but also more difficult to study and analyse the model. It will also make any result we get more specific to the given system (less generalizable to other systems)

When using networks, the following aspects or levels of details are used

Defining a network: graphs

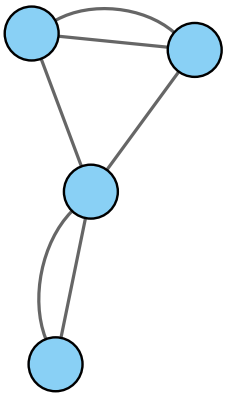


Simple graph:

- Nodes
- Edges

The most basic, bare-bones network – often called a graph (from mathematical graph theory)

Defining a network: graphs

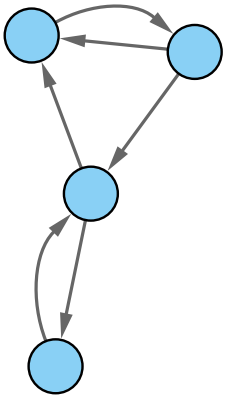


Multigraph:

- Several, parallel edges

Allowing multiple parallel edges might occasionally be needed

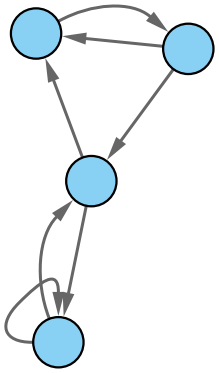
Defining a network: graphs



Directed graph

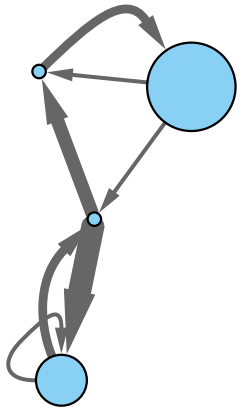
Defining a network: graphs

Self-edges



Self-edges are edges where the start and the endpoint is the same node, in this example, the node at the bottom has one

Defining a network: attributes



Additional data:

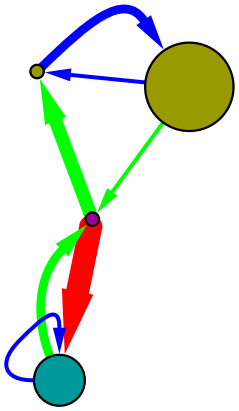
- Size / importance

In many networks, nodes and edges will be essentially similar – obviously it makes analysis much easier if they have no other property.

However, in many cases we do need to consider additional attributes for either nodes or edges – the type of this additonal data can be very varied.

Note that the image on these slides also shows example of how these can be visualized: for data that is represented by numbers (i.e. we have a ratio scale: an ordering, and we can also compare values), we can visualize as node size and edge thickness. Note that this will intuitively be interpreted by the viewer as the importance of the given elements.

Defining a network: attributes

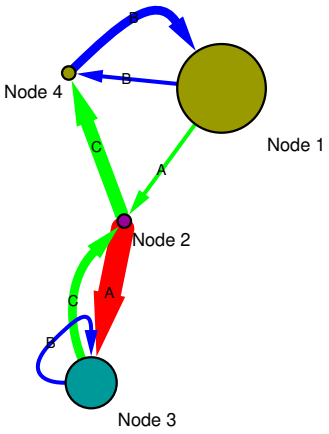


Extra data:

- Size / importance
- Type

For nominal attributes where we don’t have an ordering, we shouldn’t use a visual property that hints at an ordering. Instead, using node shape or discrete colors (not a color gradient) allows visualizing it without misunderstanding.

Defining a network: attributes



Extra data:

- Size / importance
- Type
- Anything else

Of course, any other attribute can be considered. Simply writing out the value of the attribute is not a really good way to visualize it, but it is the most versatile method for displaying it.

Bipartite network

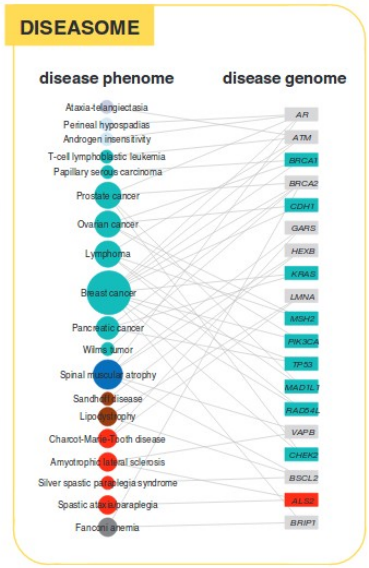


Image source: Goh KI, Cusick ME, Valle D, Childs B, Vidal M, Barabási AL. PNAS (2007)

Many networks are constructed as a bipartite network, and due to their special structure, it is worth discussing them a bit.

In bipartite networks, every node is one of two types, and edges only connect nodes of different types. An easy way to visualize it is two consider two columns of nodes, with edges only going between columns.

In this example, the edges connect medical conditions to genetic causes where a relationship has been proven.

Bipartite network and its projections

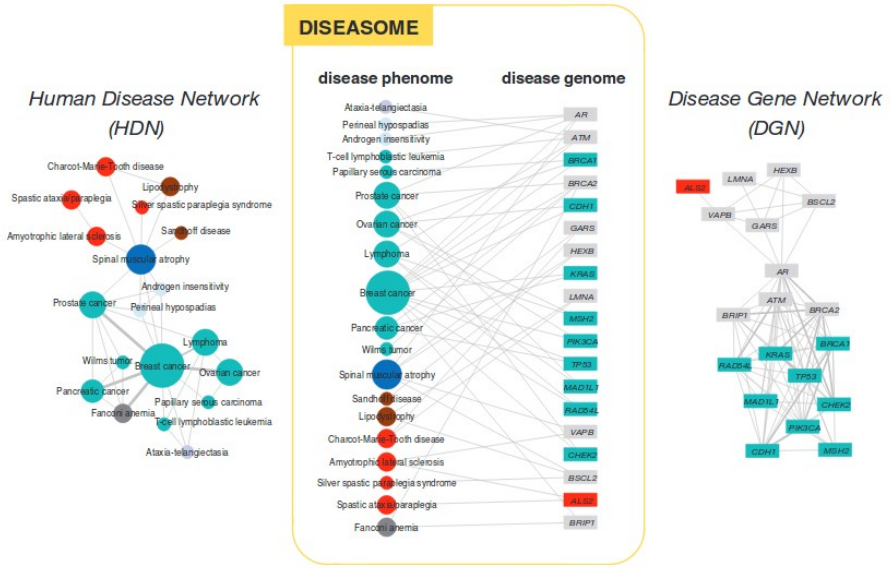


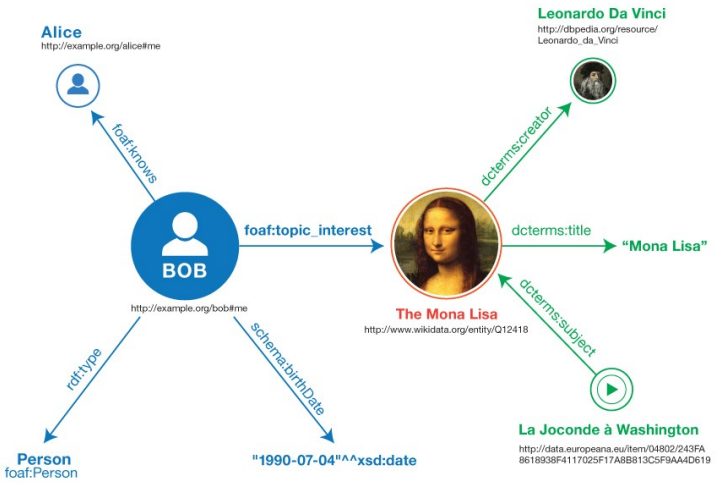
Image source: Goh KI, Cusick ME, Valle D, Childs B, Vidal M, Barabási AL. PNAS (2007)

Often, bipartite networks are not directly used, instead their projection is studied: these are formed by taking all nodes of a given type, and connecting those which have a common neighbor in the original network.

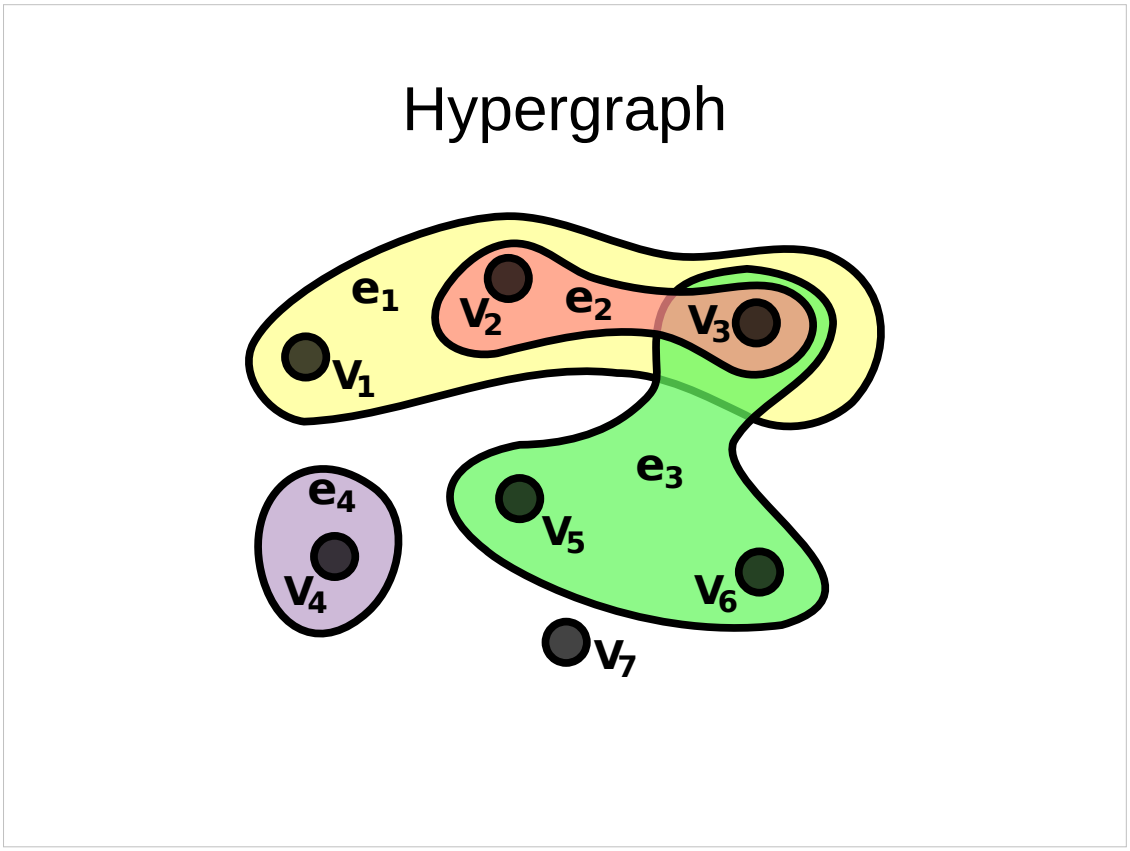
In this case, this generates a network of medical conditions which have a common genetic factor (on the left), and a network of genetic factors which are linked to the same medical conditions (on the right).

Note that this way of forming a network means that the resulting network will likely have large, fully connected cliques: for example, any genetic factor that has many connections will generate such a clique in the Disease Network.

Tripartite network: RDF



We can also consider networks of more nodetypes: one that is quite popular in certain circles is the RDF used in Semantic Web technologies, which is essentially a tripartite network. The three nodetypes are: subject, predicate, object. These are often visualized with the predicate being written on the edge, essentially as an edge type.



If we want to further generalize this idea, we will end up with a hypergraph, where edges can be any subsets of the nodes. In the example above for example, we have a “traditional” edge of e2 (orange blob) connecting v2 and v3, the “tripartite” edges e1 and e3 (yellow and green), but also an edge e4 which is connected only to v4 (we might consider this a “traditional edge” as a self-edge, or we might treat this as something else).

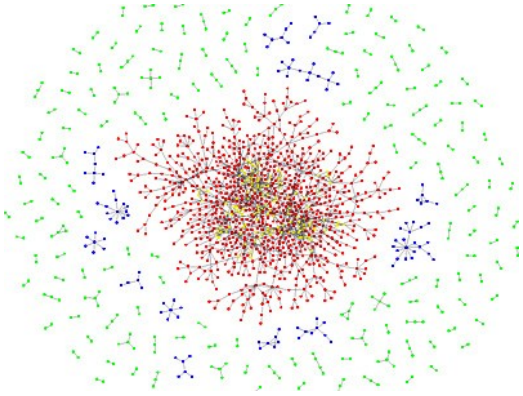
Note that such generality makes hypergraphs much less useful than traditional networks: in most applications we can use a more traditional network, and there are much more mathematical results for those, making them much more useful.

Common basic properties

As mentioned before, the surprising usefulness of network science is due to the common statistical properties of the networks resulting from widely disparate systems.

In the following, we will list some of these basic properties. Note that not all networks will have all of them. A good way to get started with analyzing a network is to check which of these properties they have.

Components



- Isolated parts
- One large connected component

→ complications for:

- shortest paths
- diameter
- any quantity defined for a component

Image: an online community network, from:
http://www.visualcomplexity.com/vc/project_details.cfm?id=139&index=139&domain

A fundamental aspect of the network is whether it is connected, i.e. whether we can reach any node from any other node via the edges.

Note that in many cases, even in systems which have isolated parts, we will only observe one connected component, if we are limited to collecting data by traversing the edges.

Many mathematical quantities are defined assuming that the network is connected, which leads to using various tricks / modifications. For example, strictly speaking, the diameter of such a network will be infinite, but that is quite useless information (since this will lead to essentially every network having an infinite diameter), so one might instead calculate the average diameter of the components.

Sparseness

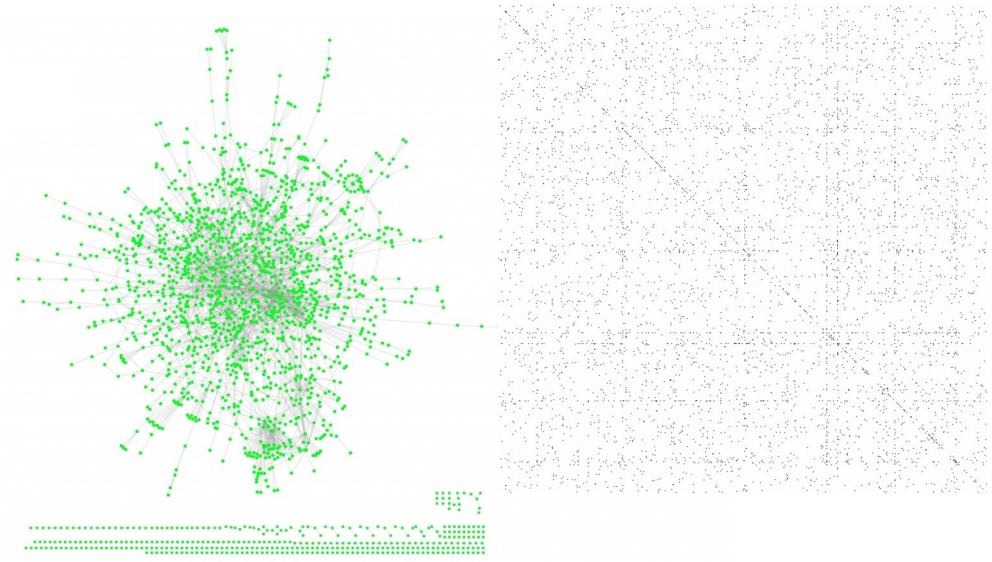
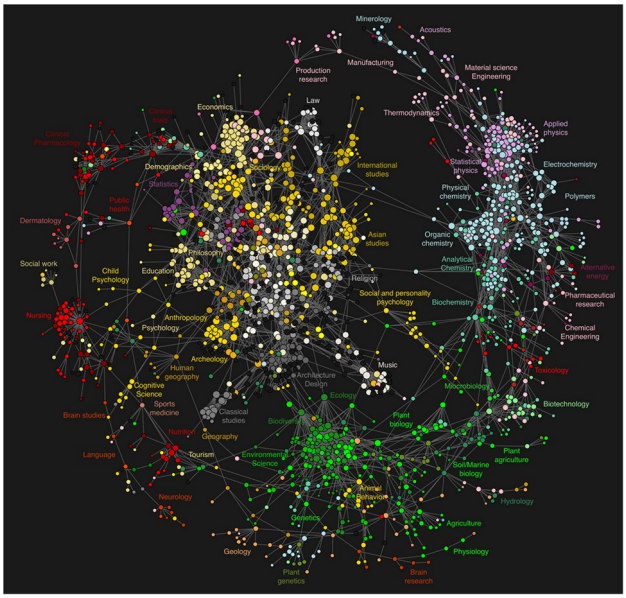


Image 2.4 and 2.7 from Barabasi's book

Most real-world networks are sparse, i.e. among the possible connections, most are not present in the network. On the left: a visualization of the network using nodes and edges, on the right: the same network as a matrix: each edge is represented as a dot: the image is mostly white due to the large number of missing connections.

Note that this can be framed instead as: most systems where network-science methods (as opposed to other mathematical methods) are useful will have networks that are sparse

Globally sparse, locally dense



Structure of science based on readership (clickstream) data of scientific articles
http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0004803

However, as seen on the previous image, parts of the network are dense: i.e. sparseness is a global property, while locally most networks are dense, with a large number of triangles, or larger cliques.

Small world property

Compared to grid

advantages and disadvantages

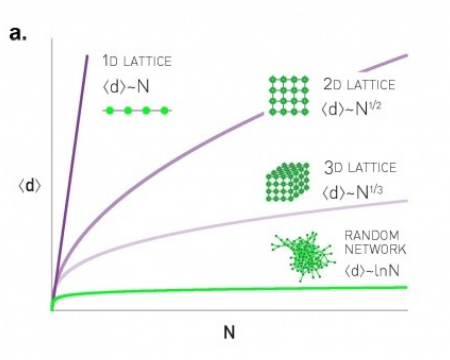


Image 3.11 from Barabasi's book

An often cited result is the small-world property, which is essentially that the diameter of the network is larger than what one would expect based on the number of nodes.

To define this in mathematical terms, one needs to vary the size of the network (the number of nodes), and talk about the diameter as a function of the number of nodes. The diameter will obviously increase as the number of nodes grows, but important part is how it increases: regular grids result in a polynomial growth (square root for 2d grid, cubic root for 3d grid), while logarithmic growth indicates a small-world property (note that this means that the number of nodes reachable in S steps grows exponentially)

Also note that this assume the network is connected: unconnected parts will result in the boring result of infinite diameter

Degree distribution

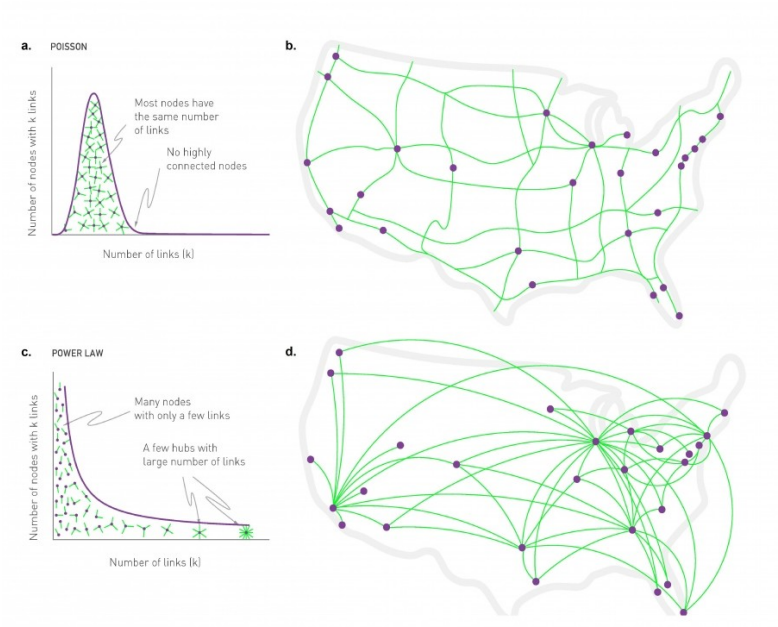
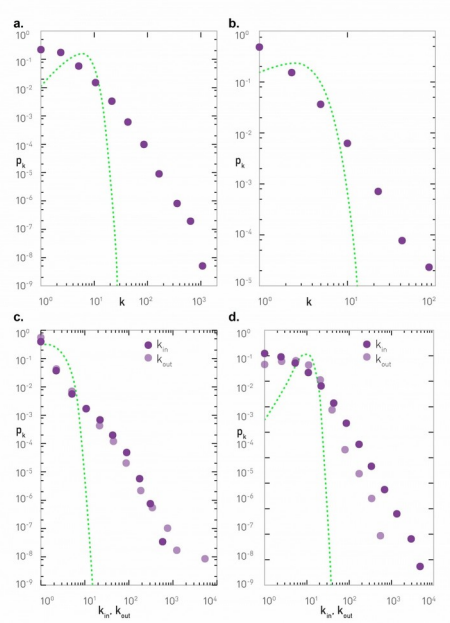


Image 4.6 from Barabasi's book

An important aspect that fundamentally affects how the network behaves is the degree distribution. Often, this distribution can be traced back to how the network was formed: for example, the airline transportation network (bottom) depends on hubs, large airports that have flights to a lot of other airports, while the road network (top) is much more like a regular grid. The physical limitations affecting the two systems are different: no point can be the meeting point of several hundred roads, but an airport can have several hundred flights a day.

Note that this difference, for example, has huge effects on the spreading of epidemics

Degree distribution



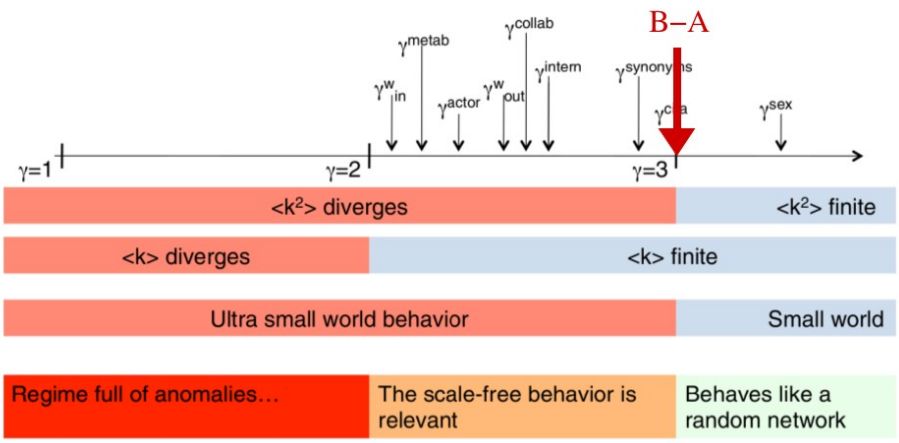
- Four networks:
  - Internet routers
  - Protein interactions
  - Email network
  - Citation network of scientific articles
- Purple dots: network
- Green line: Poisson-distribution with same <k>

Image 4.10 from Barabasi's book

Plotting the degree distributions for these four networks shows that the Poisson-distribution (which is a non-fat-tailed distribution) is a very bad fit, as these are fat-tailed distributions.

In many cases, powerlaws are used to describe these distributions.

Consequences of the distribution



From Barabási's slides

An important aspect of powerlaws is that, depending on the exponent of the powerlaw (gamma on the image), certain mathematical quantities might not actually exist.

The arrows at the top of the image mark various values of gamma that have been measured in real-world systems.

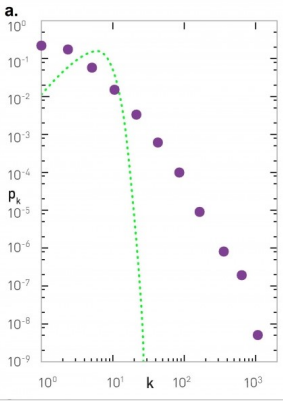
A warning

- Power law
- Scale-free
- Fat-tailed

	name	distribution $p(x) = Cf(x)$ $f(x)$	C
continuous	power law	$x^{-\alpha}$	$(\alpha-1)x_{\min}^{\alpha-1}$
	power law with cutoff	$x^{-\alpha}\mathrm{e}^{-\lambda x}$	$\frac{\lambda^{1-\alpha}}{\Gamma(1-\alpha,\lambda x_{\min})}$
	exponential	$\mathrm{e}^{-\lambda x}$	$\lambda \mathrm{e}^{\lambda x_{\min}}$
	stretched exponential	$x^{\beta-1}\mathrm{e}^{-\lambda x^{\beta}}$	$\beta\lambda \mathrm{e}^{\lambda x_{\min}^{\beta}}$
	log-normal	$\frac{1}{x}\exp\left[-\frac{(\ln x-\mu)^2}{2\sigma^2}\right]$	$\sqrt{\frac{2}{\pi\sigma^2}}\left[\mathrm{erfc}\left(\frac{\ln x_{\min}-\mu}{\sqrt{2}\sigma}\right)\right]^{-1}$
discrete	power law	$x^{-\alpha}$	$1/\zeta(\alpha, x_{\min})$
	Yule distribution	$\frac{\Gamma(x)}{\Gamma(x+\alpha)}$	$(\alpha-1)\frac{\Gamma(x_{\min}+\alpha-1)}{\Gamma(x_{\min})}$
	exponential	$\mathrm{e}^{-\lambda x}$	$(1-\mathrm{e}^{-\lambda})\mathrm{e}^{\lambda x_{\min}}$
	Poisson	$\mu^x/x!$	$\left[\mathrm{e}^{\mu}-\sum_{k=0}^{x_{\min}-1}\frac{\mu^k}{k!}\right]^{-1}$

TABLE 2.1

Table from:
http://tuvalu.santafe.edu/~aaronc/powerlaws/



An important warning: when talking about powerlaws, make sure you are actually talking about powerlaws. This table shows various other distributions that will appear to be very similar, and thus are easy to confuse with powerlaws.

The link on the slide is to a good publication describing the recommended statistical method to determine whether a given distribution is a powerlaw. In python, the “powerlaw” module is good and easy-to-use implementation of that method.

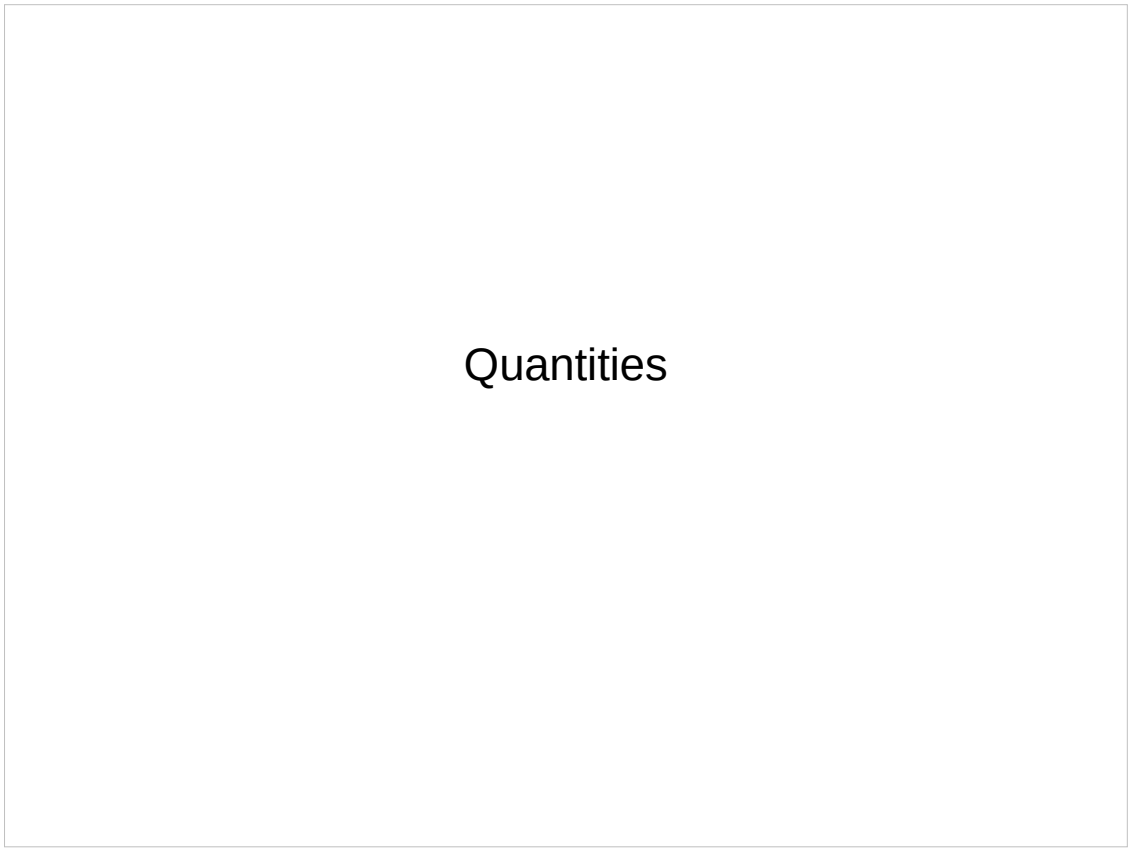
Also note that in many cases, one can avoid this issue by say “fat-tailed” instead of “powerlaw” or “scale-free”: if all that is needed is that whether there are hubs in the given network, that can be determined just by looking a the distribution, without any further statistical proof.

What wasn’t visible on these slides

- Dirty data, artifacts
- “you get what you pay for”
- Network analysis is more sensitive to data cleaning than other methods
- But: for data analysis, less of a difference between “artifact” and “interesting phenomena”

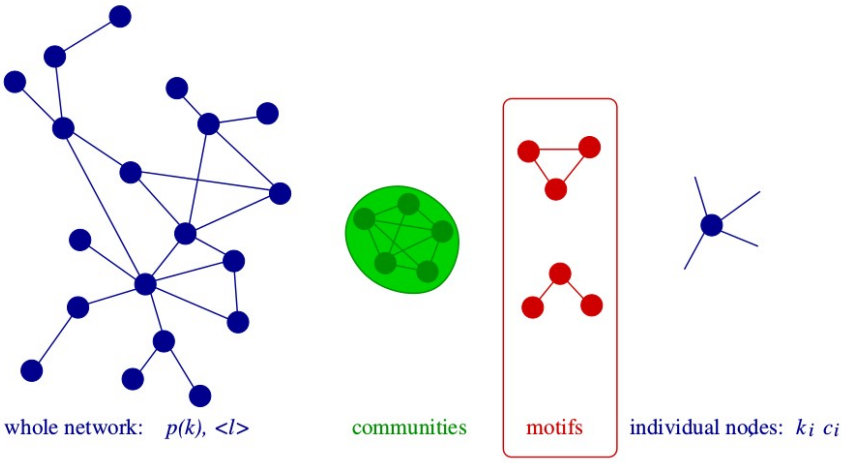
The last few slides shows networks exhibiting various properties of the underlying systems. In most real-world datasets, many of the properties that you might detect are actually artifacts. Most network datasets are not collected to do data science, but essentially accidentally result from some other investigation. This means that the validity or completeness or integrity of the data is often lacking. Since network science can use global properties which can be changed by small local changes, it is usually more sensitive than other methods.

Note that in terms of data analysis, it makes no difference whether a given pattern is due to some property of the original system or is an artifact of the measurement: the only difference between the two is the explanation we use for it, but explanations should be considered a further step, beyond data analysis.



There are very many different mathematical quantities we can calculate. In the following, my main aim is to give only an overview of these, and I will mostly focus on a giving you a framework for organizing them.

Scales



One useful aspect to consider is the scale of the given quantity: how large a part of the network do you need to consider to calculate it?

Note that one way to get a “whole network” quantity is to define the distribution of some smaller-scaled quantity: thus, the degree distribution is such, even though degree itself can be calculated based only on individual nodes.

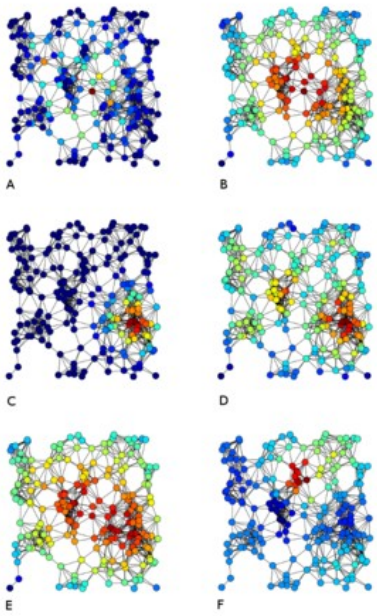
Quantities

- Degree – directed (in- & out-), weighted (strength)
- Edges → Degree correlation (assortativity)
- Triangles → clustering coefficient
- Shortest paths → betweeness
  - Geodetic vs. random paths
- Whole network – degree, distribution

Listing some more examples, starting from one single node, as we use more and more of the network, we can define these quantities.

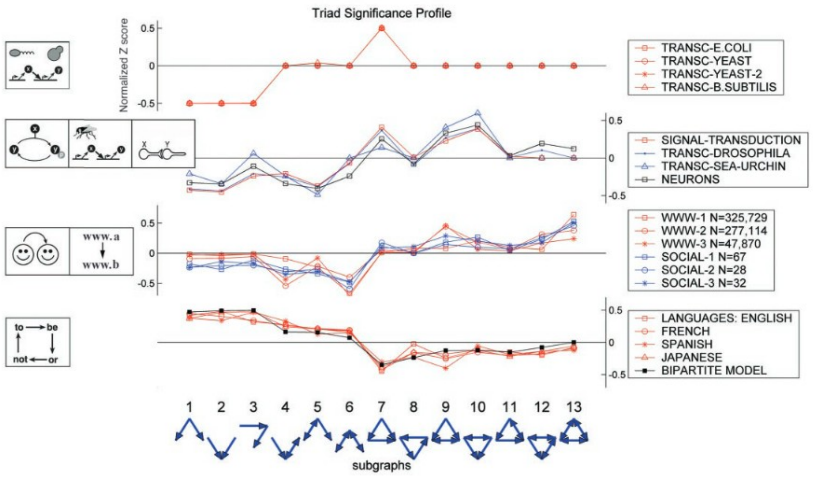
Various centrality measures

- A) Betweenness centrality
- B) Closeness centrality
- C) Eigenvector centrality
- D) Degree centrality
- E) Harmonic centrality
- F) Katz centrality



Many quantities result in a number for each node. Depending on the meaning of the quantity, these can be considered the importance or centrality of the node (using various ways of defining importance)

Motifs



Milo, R; Itzkovitz, S; Kashtan, N; Levitt, R; Shen-Orr, S; Ayzenshtat, I; Sheffer, M; Alon, U
Superfamilies of Evolved and Designed Networks
Science , (2004)

Motifs allow defining a “fingerprint” of the network: the relative frequency of these small patterns (for example triangles, but one can use larger sub-graphs as well) will be affected by the structure and functioning of the network.

On this image, these relative frequencies are plotted, as the different curves show, the four type of networks can be clearly distinguished.

The high relative frequency in some cases can be explained, for example it might be due to the triangle-closing property in the dynamics of social networks, or the importance of the noise-filtering property of the feed-forward loop for gene regulation networks.

Random models

Models in general

- Qualitative model: what are the fundamental mechanisms?
- Quantitative model: act as reference

When discussing models, it is important to keep in mind that models can be used for two very different purpose: a qualitative model is used for understanding basic behavior and thus needs to be simple, while a quantitative model needs to be detailed enough such that accurate numerical results can be calculated and also so that the parameters can be fitted to the system being modeled.

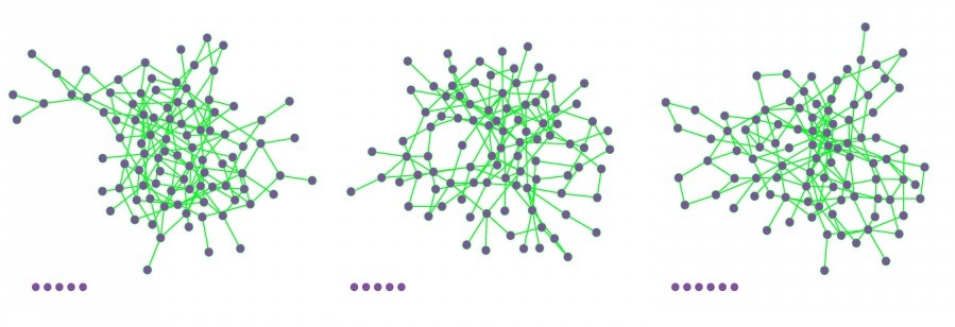
Note that this distinction is not binary, instead it is a continuum: one can usually define more and more detailed models which will be more and more quantitative, or simplify a given model by removing certain aspects or details thus moving towards a more qualitative model.

Network science usually focuses on qualitative models.

Erdős-Rényi model

- N nodes and:
  - M edges

    or:
  - Every pair of nodes connected with p probability

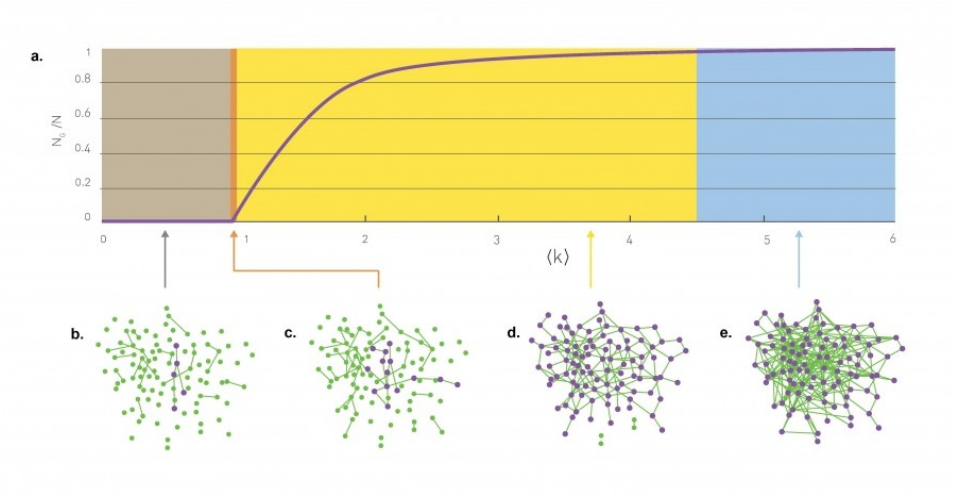


N=100, p=0.03 – Image 3.3 from Barabási's book

The simplest possible model, where we simply have a set amount of nodes and edges (or a set amount of nodes, with a given probability of connecting them)

Erdős-Rényi model

- If adding edges one by one: percolation

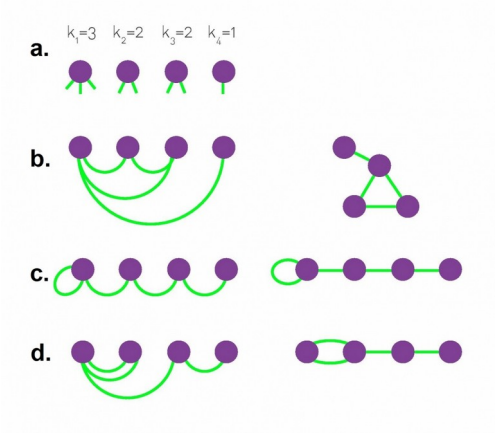


Increasing the number of connections (or alternatively, increasing the probability of the connections) changes the structure of the network, in a process that is often called percolation:

Below a critical point, in the brown regime, the network is made up of small, isolated components. At the critical point, a giant component starts to form. This is tree-like close to the critical point, and grows as we go towards the right on this image.

The dark curve shows the size of the largest component, as a fraction of all nodes.

Configuration model: given degrees



- Nodes & edge ends
- Connecting edge ends
- (not all sequences suitable)

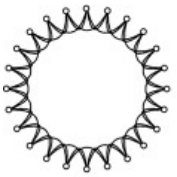
Image 4.15 from Barabási's book

If we go beyond determining the number of nodes and edges, we can also consider specifying the degree sequence (which also determines the degree distribution). The configuration model accomplishes this, and is often constructed by considering isolated edge-ends, which are then joined together.

On the image, a) is the starting edge-ends, b,c,d are various ways they can be connected: the column on the left uses the same node placement as for a), while the column on the right shows a re-arranged network, which shows the structure of the network better.

Watts-Strogatz – small world

- Ring, close neighbors connected



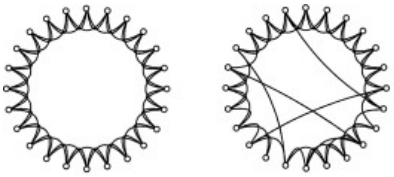
A significant limitation of both the Erdős-Rényi and the configuration models is that the node's clustering coefficient will be low, as the probability of forming triangles will be low. We have seen that many real-world networks are locally dense, which means also means that they will have high clustering coefficient.

An early model that can produce high clustering coefficient is the Watts-Strogatz network, which is also a good example for a network model which essentially interpolates between two models.

It starts with a ring of nodes, with neighbors (up to a few steps, say first and second neighbors) connected. This can be considered as a form of a regular grid.

Watts-Strogatz – small world

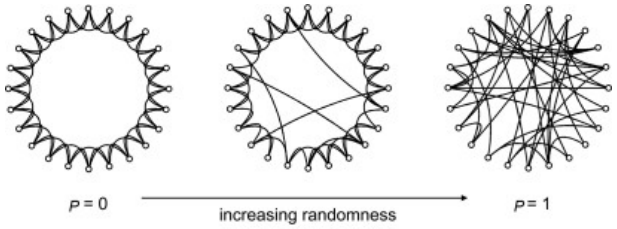
- Ring, close neighbors connected
- Randomly move edges



Then, edges are moved at random, with the number of edges being moved (or the probability of moving any given edge) being a parameter used to control how randomized the network is.

Watts-Strogatz – small world

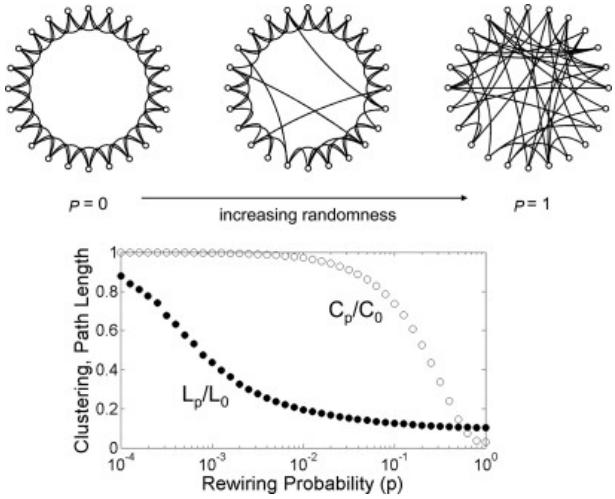
- Ring, close neighbors connected
- Randomly move edges
- At end: essentially Erdős-Rényi network



Obviously, if we move all the edges, we essentially end up with a fully random network, I.e an Erdős-Rényi network

Watts-Strogatz – small world

- Ring, close neighbors connected
- Randomly move edges
- At end: essentially Erdős-Rényi network
- Interim state: high clustering, low diameter



The important behaviour, however, will be between these two extremes: plotting the clustering coefficient and the diameter of the network we can see that there is going to be a range in the middle where the network has both high clustering (since most of the edges were not touched and the clustering was high in the original grid), and also has low diameter (since a few shortcuts are enough to achieve a large drop in the diameter).

Barabási-Albert model

Growth process– preferential attachment

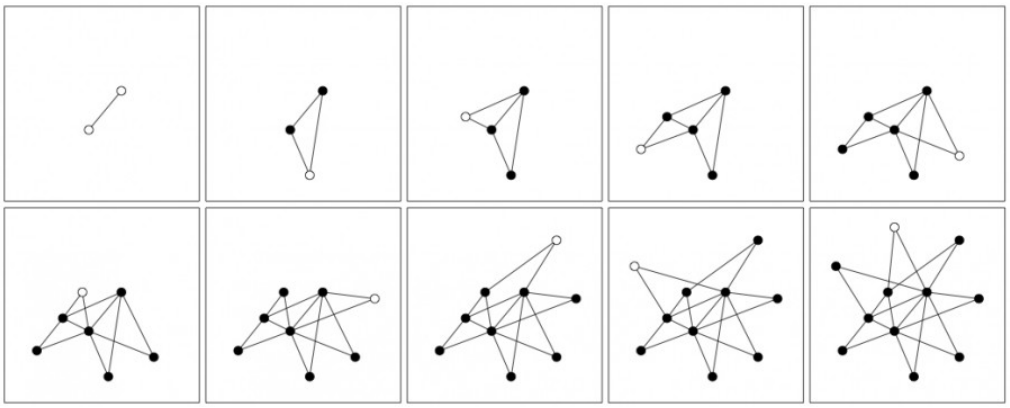


Image 5.3 from Barabási's book

Another popular way to formulate a model is to describe a growing process. This also allows generating networks of various sizes, and many theoretical arguments become easier as larger and larger networks can be used, or "N goes to infinity" asymptotics can be studied.

A very famous model is the Barabási-Albert model, where the main feature is that the growing process is preferential: new nodes are added at each timestep, which are connected to existing nodes. These connections preferentially pick nodes of high degree. This means that nodes that have high degree are more likely to get new edges and thus get even higher degree.

This model leads to a powerlaw degree distribution.

The dynamic itself also appears in many systems: for example when starting a new airline, flying from existing hubs might help getting customers.

Random models – how to use them?

- Consider parameters
- Use as large network as possible
- Average over multiple runs
  (but one might be enough)

"create from scratch"

vs.

"randomizing existing data"

Some general notes about using the models:

Make sure to pick the correct values for the parameters. For example, if a reference model is needed for an existing network, make sure to use the same number of nodes, number of edges, etc. to avoid differences in these causing a difference in the result.

If possible, use large networks since small networks often have finite-size effects obscuring the basic properties.

Make sure to generate use a sample of networks, to avoid random noise or stochasticity affecting the result. Note that the random noise will, in general, be smaller for larger networks, so a smaller sample is enough for larger networks.

Also keep in mind an alternative method: instead of generating a random network from scratch, using the existing data and randomizing it (for example rewiring edges) can be a good reference, too.

Clustering / Community finding

- From “globally sparse, locally dense” property → find dense sub-sets of nodes
- Modules / Clusters / Communities
  - Different names for the same general idea
  - (note: “clustering” in this case has nothing to do with “clustering coefficient”!)
- Very under-defined problem, hard to measure accuracy
  - Everyone will have their favorite definition / method
- Very helpful for visualization
  - Can use for high-level overview
- Many methods from other disciplines:
  - Data mining, unsupervised learning, statistics, etc.

Varieties of definitions of groups

- Disjunct: each node placed in only one group
- Fuzzy: each node is in every group, but with different weights
- Overlapping: each node might be in several groups
- Hierarchical: groups defined at various levels

And also: is every node placed in a group, or might some be left out?

There is even disagreement on how the result of a clustering should look like: what are the groups like?

Most method use the “distinct” definition, since it is easiest to handle, but many usecases require a more complicated structure.

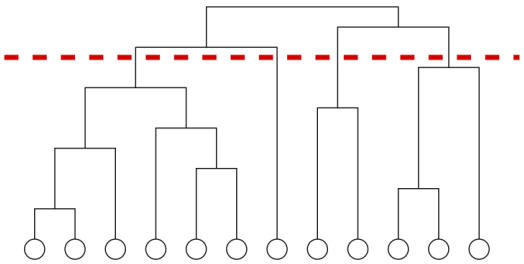
Agglomerative / divisive methods

- Agglomerative: start with isolated nodes, consider each node a separate community, at each step join two communities
- Divisive: start with one single community, at each step divide one community into two
- Easy to define greedy algorithms
- An example for divisive method: (Girvan-Newman algorithm) cut edges by always removing the edge with the highest betweenness
- Note: need separate criteria for when to stop

Some examples / types of methods

Dendrogram

- Steps of joining / separating communities
- Individual nodes at the bottom
- Final structure: horizontal cut at determined height



When joining or dividing communities, we can build a tree of these communities, to visualize how the algorithm proceeds. This is called a dendrogram.

Optimizing a function

- If a single score can be defined for the goodness of the communities, we can optimize that
- Most popular variant: Modularity:

  Compares number of edges within / between communities to those expected based on configuration model

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w) = \sum_{i=1}^{c} (e_{ij} - a_i^2)$$

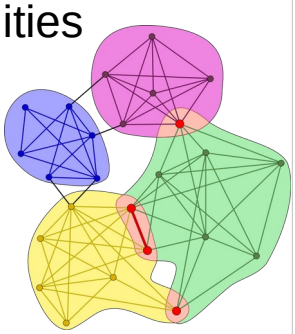
If we can build a score that measures how good a given clustering is, we can use various methods to optimize that (eg. simulated annealing, genetic algorithms, brute-force search, etc.)

The most popular such score is modularity (often denoted by Q) – for details, see the wikipedia page at https://en.wikipedia.org/wiki/Modularity_(networks)

Note that this definition has various issues, for example a resolution limit.

Clique percolation method

- Use k-clique (fully connected network of k nodes)
- Community is a part of network where clique can be moved to by moving one node at a time (“rolling the clique”)
- Allows distinct, overlapping communities



Software & tools

Cytoscape

- GUI program, point & click interface
  - Introductory tutorials: https://github.com/cytoscape/cytoscape-tutorials/wiki
- Very good for assembling visualizations
- Large number of plugins: http://apps.cytoscape.org/
- Drawbacks:
  - Gui program, so scripting is problematic
  - Performance limitations for large networks

The main strength of cytoscape is the vizmap interface, which allow interactive configuration of the visualization by mapping attributes (i.e. data) to visual properties (i.e. certain aspects of the visualization, like node size or edge color)

networkx

- https://networkx.github.io/
- Very user-friendly library for python
- Many algorithms implemented
- Drawbacks:
  - Configuring visualization more fiddly than with cytoscape (gui style editor vs. writing python code)
  - Not performance oriented, other libs will work better for huge networks

For writing quick scripts to do network data analysis, networkx is a very good choice.

Other tools

- igraph – http://igraph.org/
  - C library, R, C++ and python wrapper
  - Harder to use than networkx, but more optimized for performance
- Gephi – http://gephi.org
  - Very similar to cytoscape (Java desktop app, plugin-system)
- Neo4j – http://neo4j.org
  - Graph database, not “graph compute engine”
  - Not needed for small data
- Distributed parallel computation systems
  - Graphx (https://spark.apache.org/graphx/)
  - Giraph (http://giraph.apache.org/)
  - Etc.

Of course, there are very many other options.