Introduction to Graph Theory

Proteomes Interactomes and Biological Networks

November 19, 20 and 26, 2019

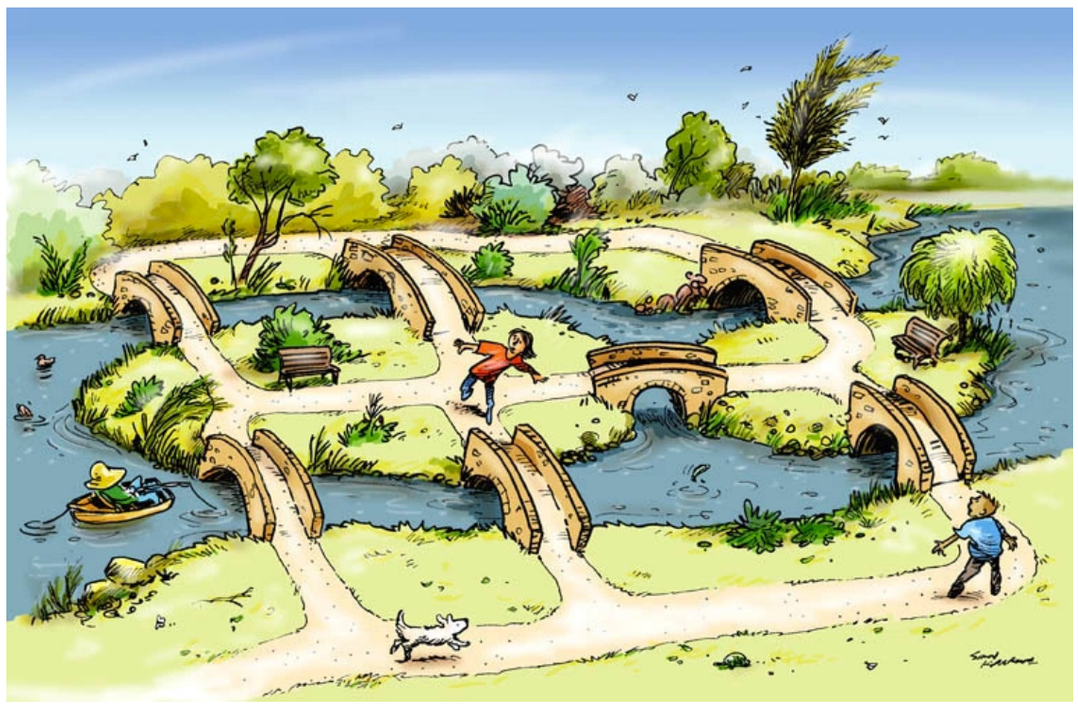
Emidio Capriotti
http://biofold.org/

Biomolecules Folding and Disease

Department of Pharmacy and Biotechnology (FaBiT)
University of Bologna

Historical Perspective

With the Seven Bridges of Königsberg problem, Euler in 1737 laid the foundations of the graph theory.

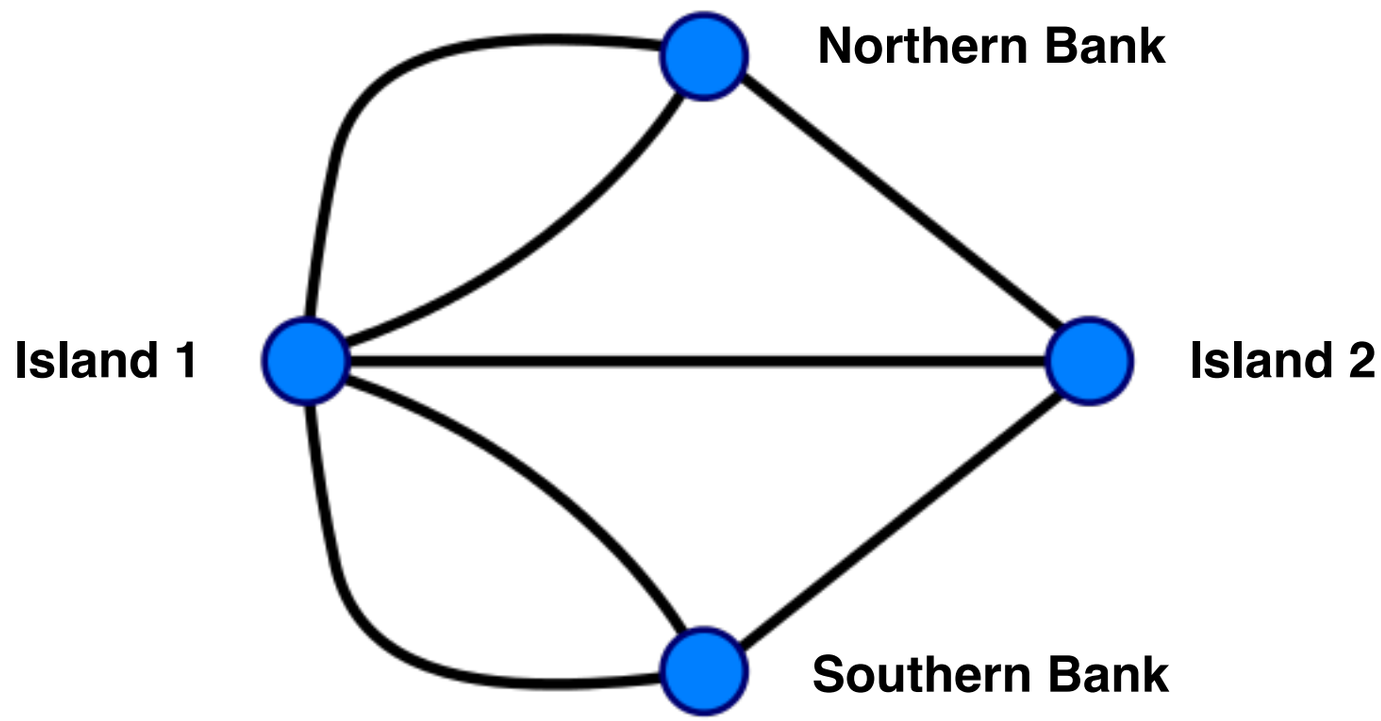


Simon Kneebone – simonkneebone.com

- Find path (Eulerian Path) that traverses all the Pregel's bridges.
- Find walk (Eulerian Circuit) that traverses all the Pregel's bridges and has the same starting and ending point.

Solution

Describe the problem as a graph where the nodes represent the 4 locations and the edges correspond to the bridges



Eulerian path exists only if zero or 2 nodes are connected by an odd number of bridges.

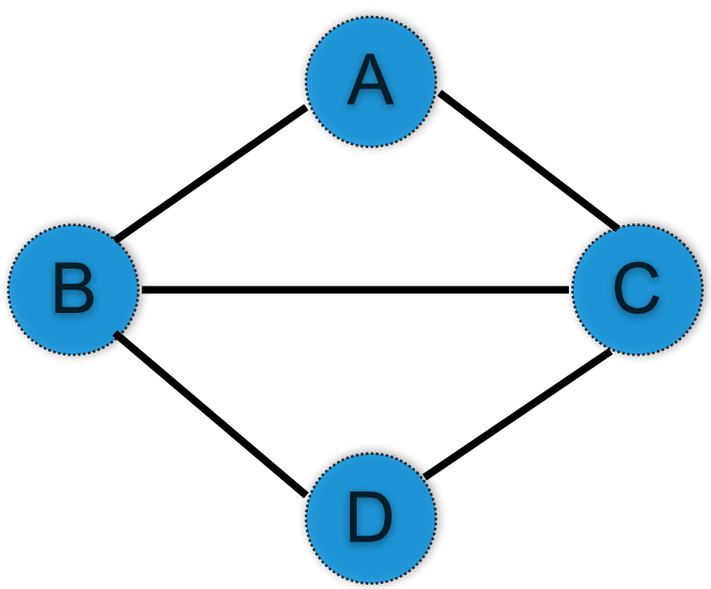
Eulerian circuit exists only if zero nodes are connected by an odd number of bridges.

Graph Definition

A graph is a pair G=(V,E) consisting of two sets:

- V is a set of elements called Nodes or Vertices.
- E is a set of pairs (v_i,v_j) where $v_i \in V$ and $v_j \in V$.

The pairs E are links between two nodes and are called Edges

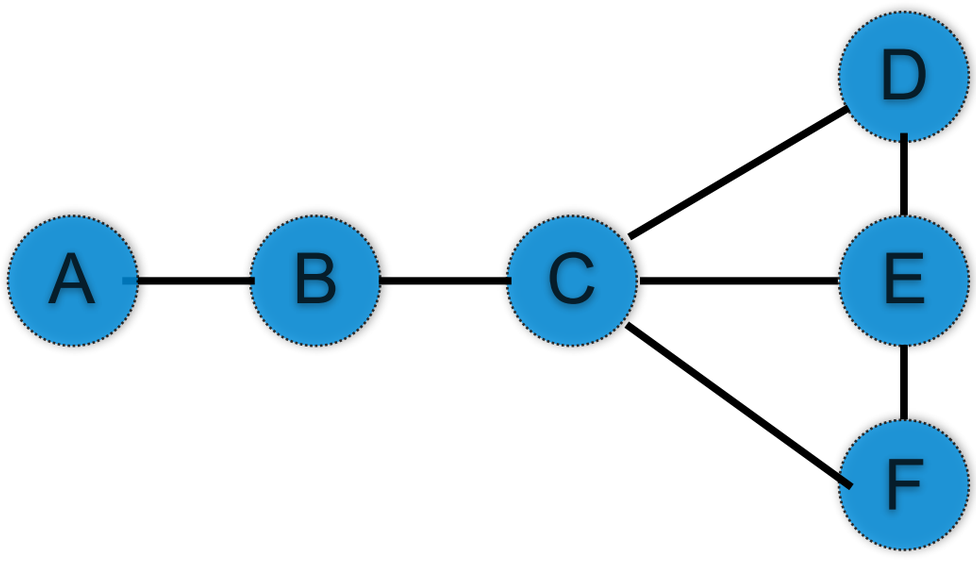


V = {A; B; C; D}

E = {(A,B); (A,C); (B,C); (B,D); (C,D)}

Undirected Graph

Undirected graph is a network where the relationship between nodes are symmetric.

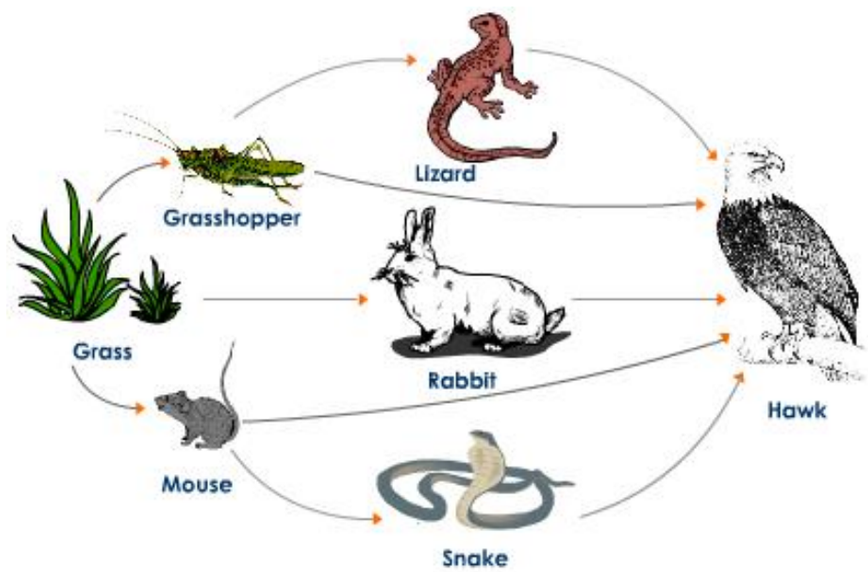


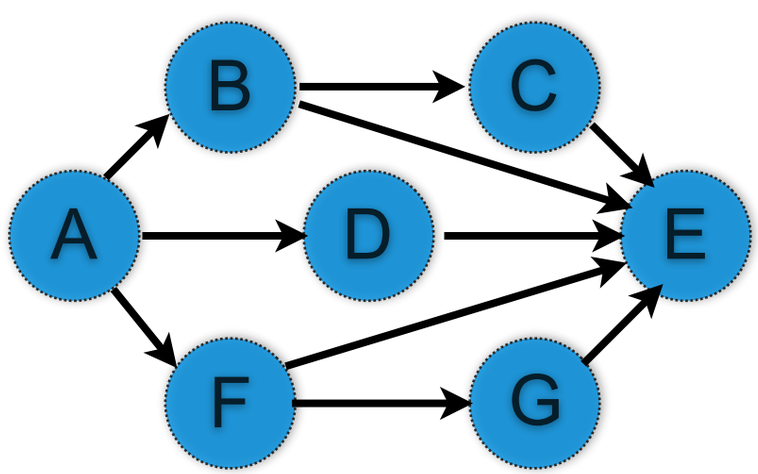
V = {Group of People}

E = {Pairs of Friends}

Directed Graph

Directed graph is a network where the relationship between nodes are asymmetric. In this case the edges are directed lines.



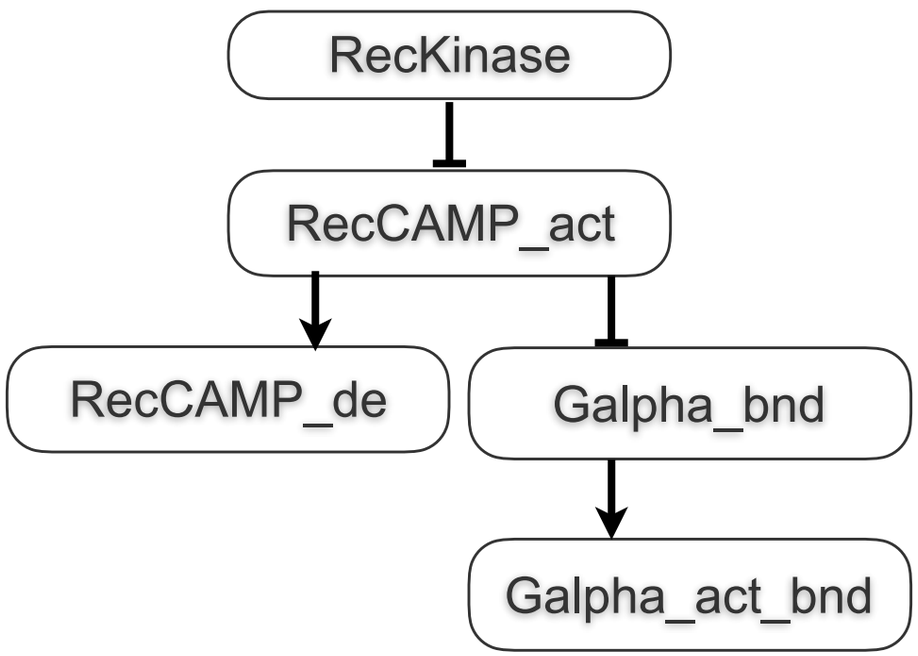


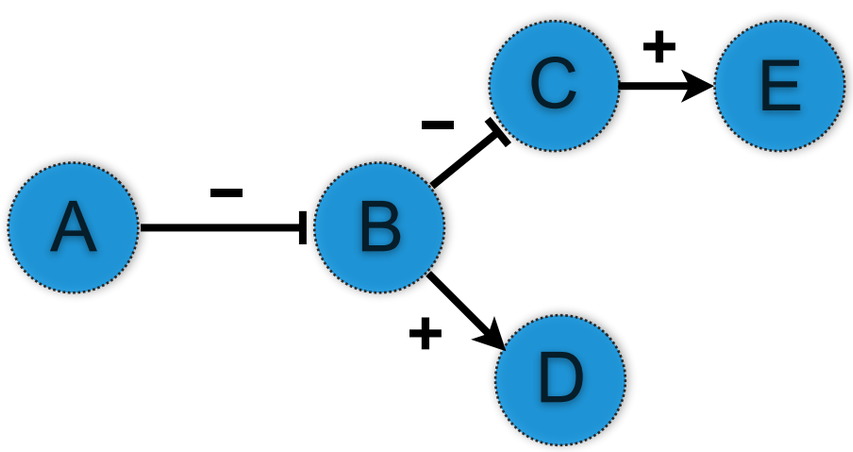
V = {Group of Animals}

E = {Pray/Predator Relationships}

Signed Directed Graph

Signed Directed graph is a network where the relationship between nodes are asymmetric and have positive or negative associated signs





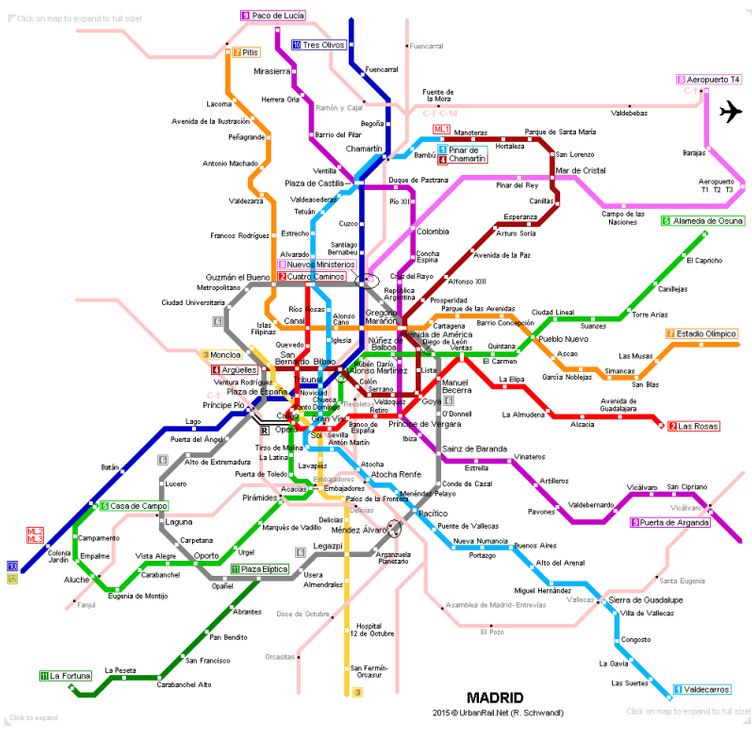
V = {Group of Genes}

E = {Activation/Inhibition Relationships}

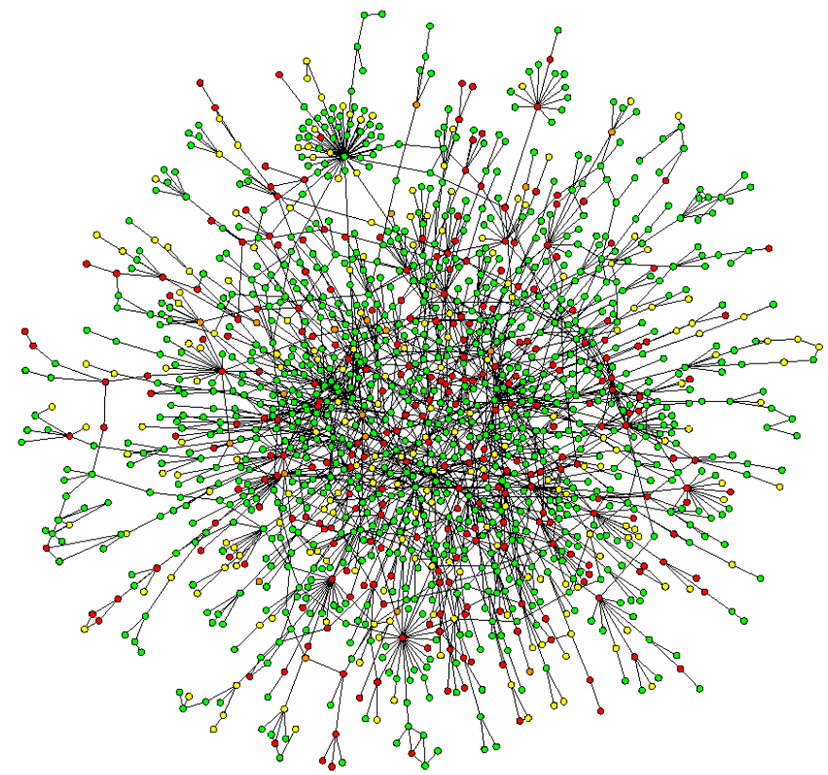
Graph and Networks

Graphs can be used to represent any observed network.

Networks in nature tend to be highly complex

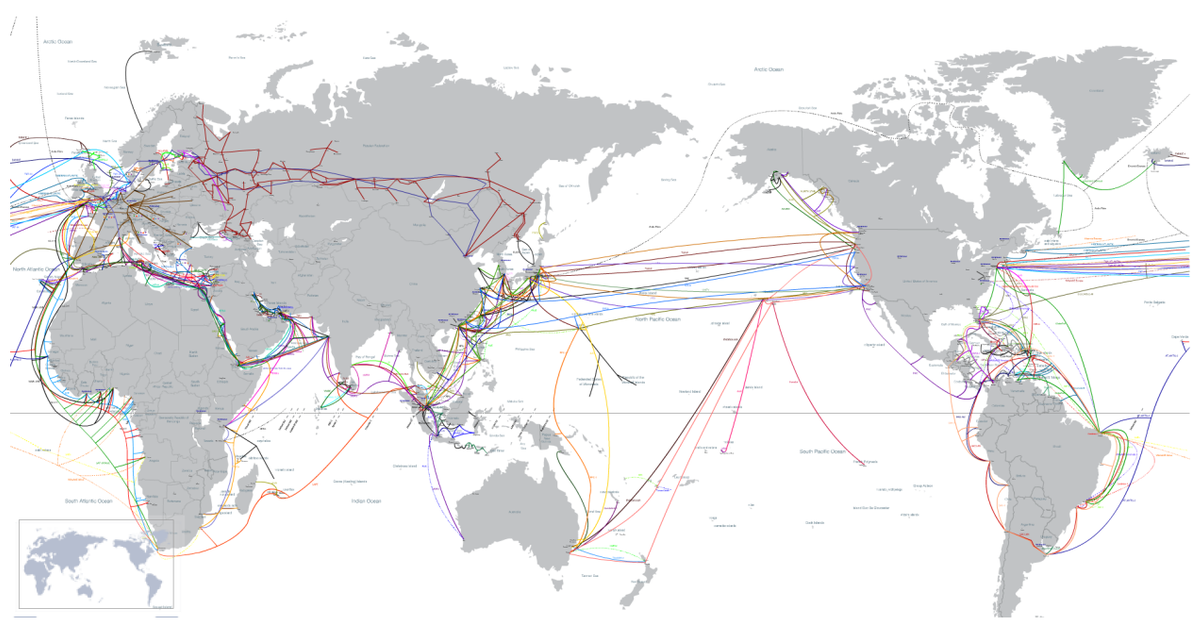


Madrid Metro



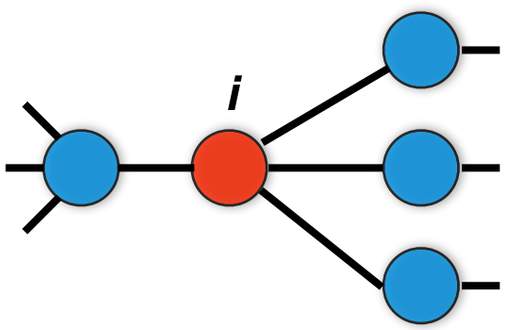
Yeast interactome

Internet connections

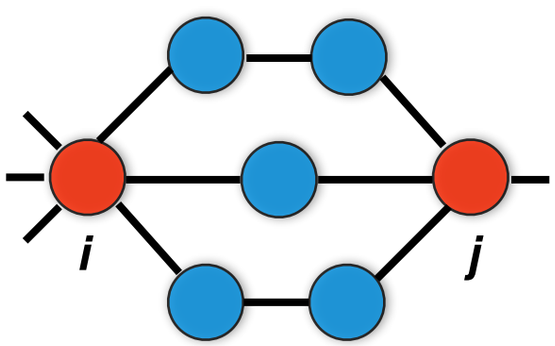


Network properties (I)

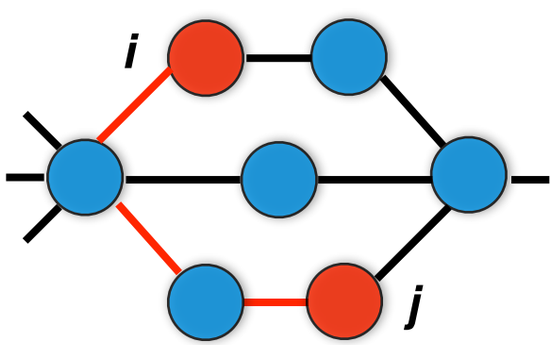
The topology of the network defines its properties. The level of connectivity among the nodes depends on the number of edges.



Degree k_i = number of links connected to node i



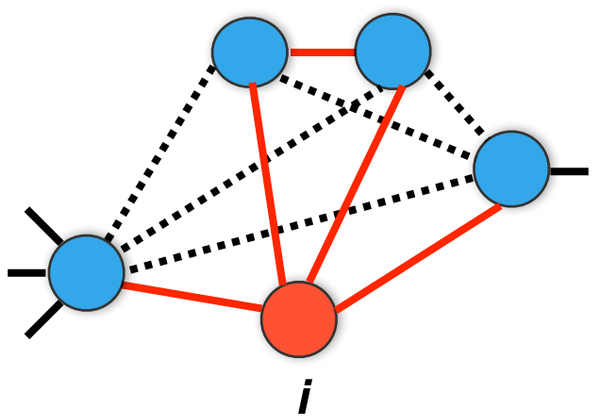
Distance d_{ij} = shortest path between nodes i and j



Diameter D = longest path between all pairs of nodes

Network properties (II)

The topology of the network defines its properties. The level of connectivity among the nodes depends on the number of edges.

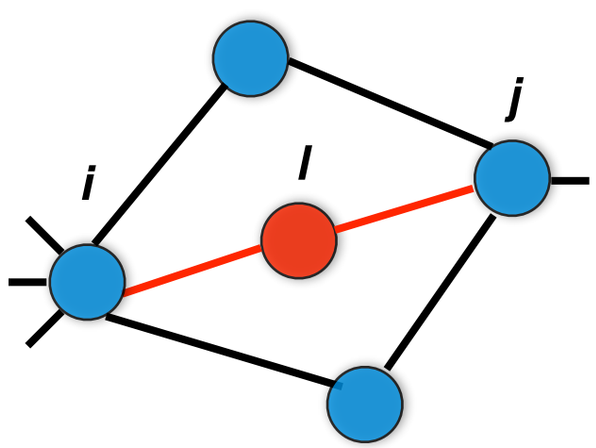


Transitivity or Clustering Coefficient

$$c_i = \frac{2e_i}{k_i(k_i - 1)}$$

k_i = number of nodes connected to *i*

e_i = number of edges between the k_i nodes



Betweenness

$$g_l = \sum_{i \neq l \neq j} \frac{\sigma_{ij}(l)}{\sigma_{ij}}$$

σ_{in} = number of shortest path between *i and j*

σ_{ij}(l) = number of shortest path passing through node *l*

Types of Network

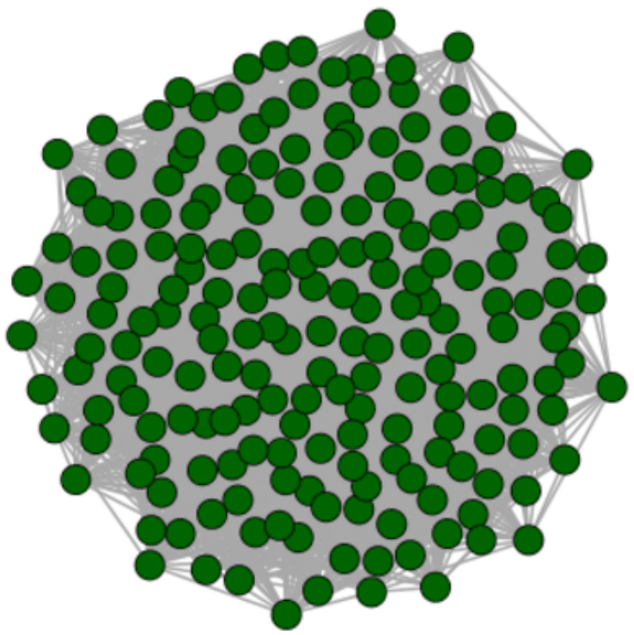
The topology of the network depends on the distribution of the degree for all the nodes.

We can define three types of network:

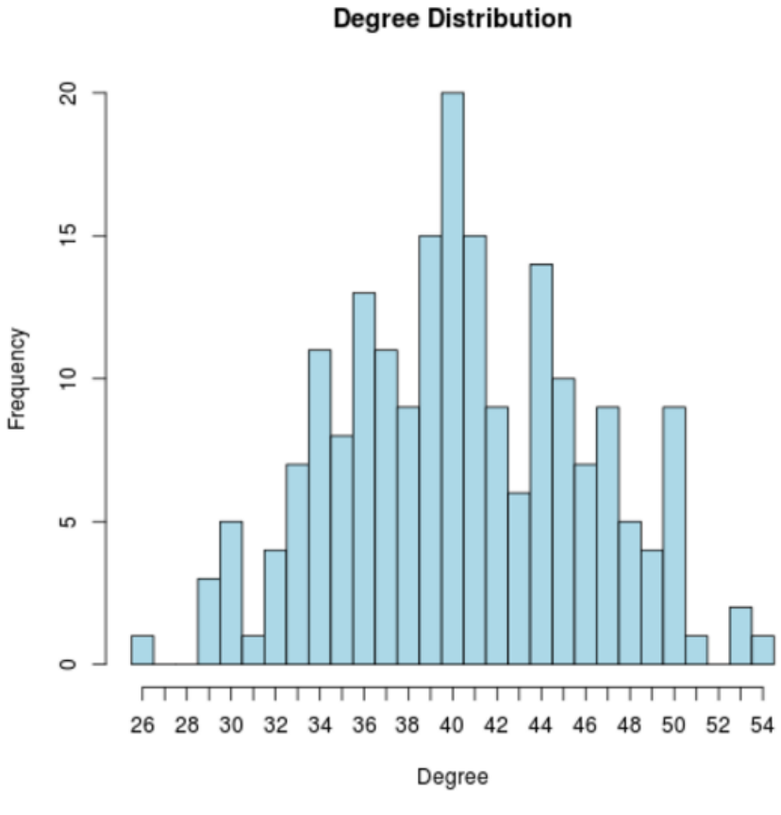
- Random network: generated by a constant probability of having a edge between two nodes.
- Small-world network: when the degrees follow a Poisson distribution
- Scale-Free network: the degrees follow a Power Law distribution

Random Network

Can be generated by Erdős–Rényi model which assume a constant probability of generating edges between nodes.



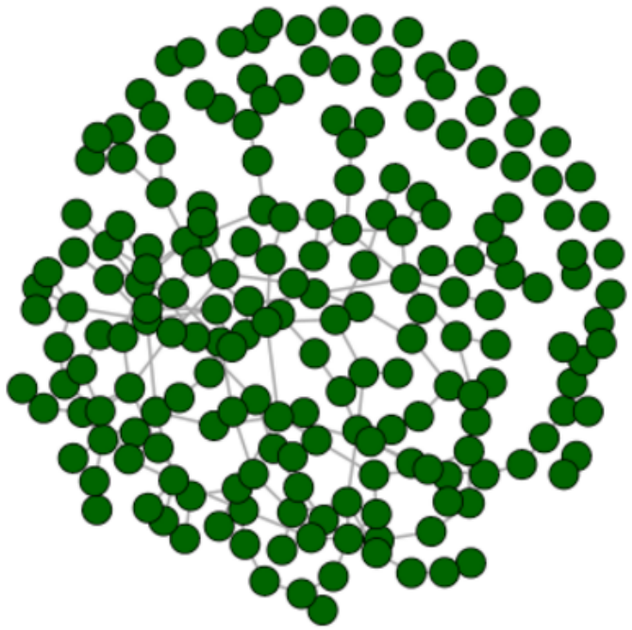
- High node degree ⇒ low average path length
- Degree distribution tends to be a Gaussian
- High Transitivity
- Small Betweenness



Degree = 40.3
Transitivity = 0.2
Betweenness = 79.3

Small-World Network

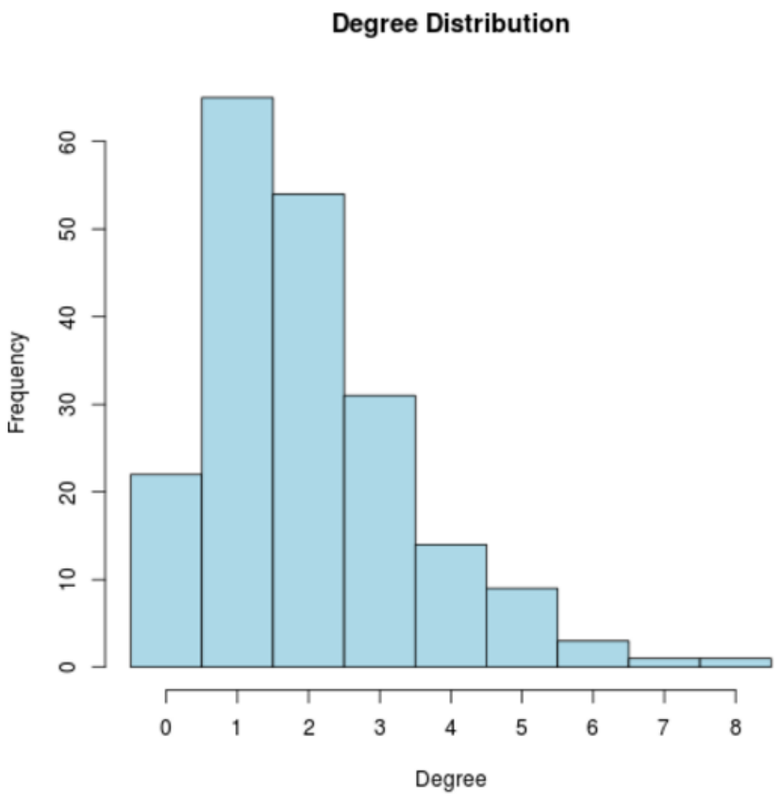
Generated by a Watts–Strogatz model.



- Low node degree ⇒ “Six degrees of separation”
- Degree follow a Poisson distribution
- Low Transitivity than random
- Higher betweenness than random

$$p(k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

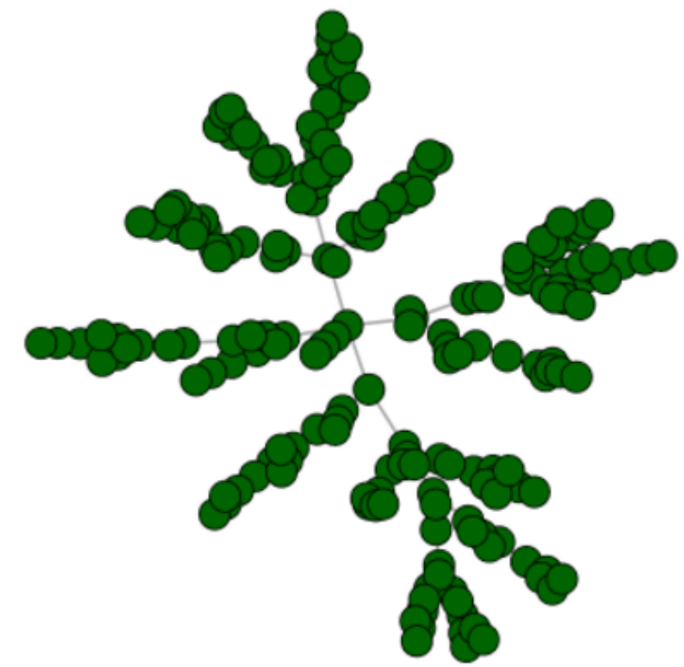
λ = the average value of the distribution
k = number of observed events



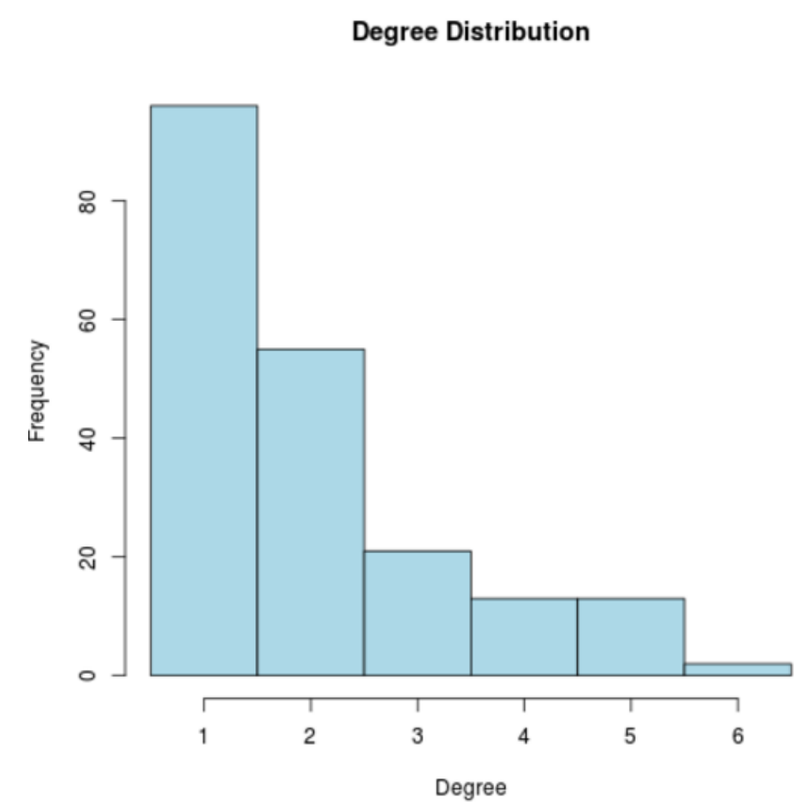
Degree = 2
Transitivity = 0.01
Betweenness = 394.9

Scale-Free Network

Generated by the Barabasi-Albert model.



- Smallest degree
- Degree follow a Power Law distribution
- Lowest Transitivity
- Highest Betweenness



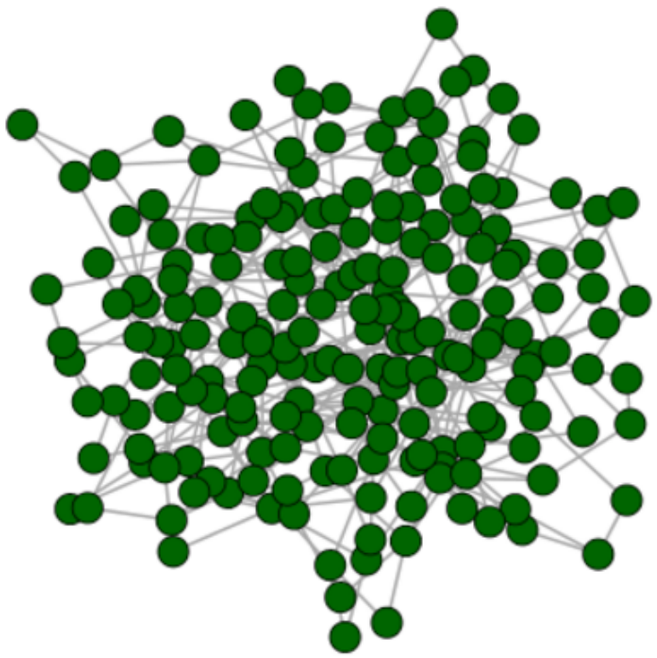
$$p(k) = Ax^{-k}$$

x = is a constant
k = number of observed events

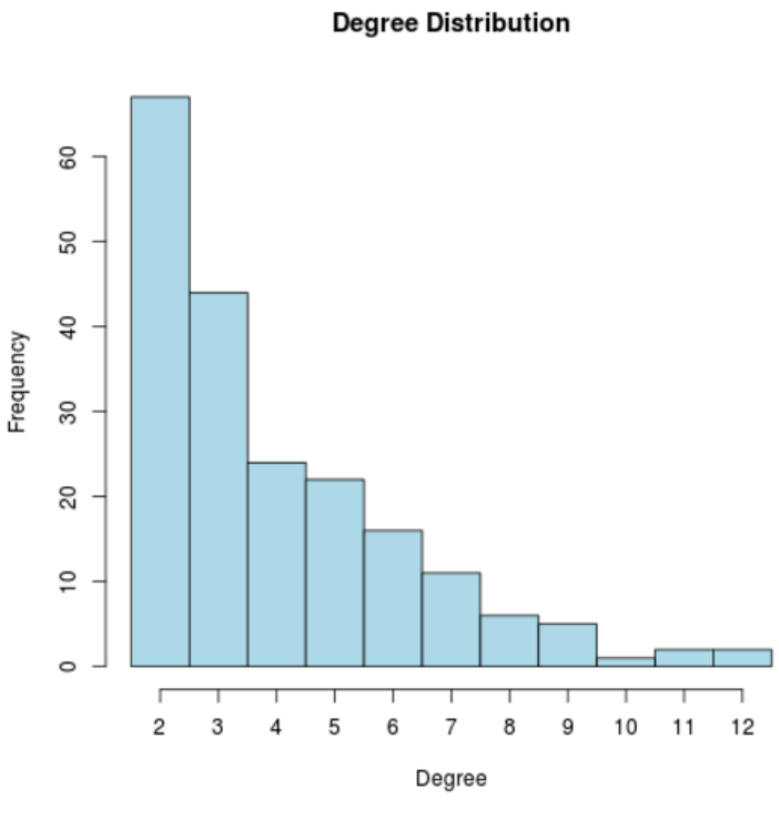
Degree = 2
Transitivity = 0
Betweenness = 753.4

Biological Network

Similar to Small-World and Scale-Free networks



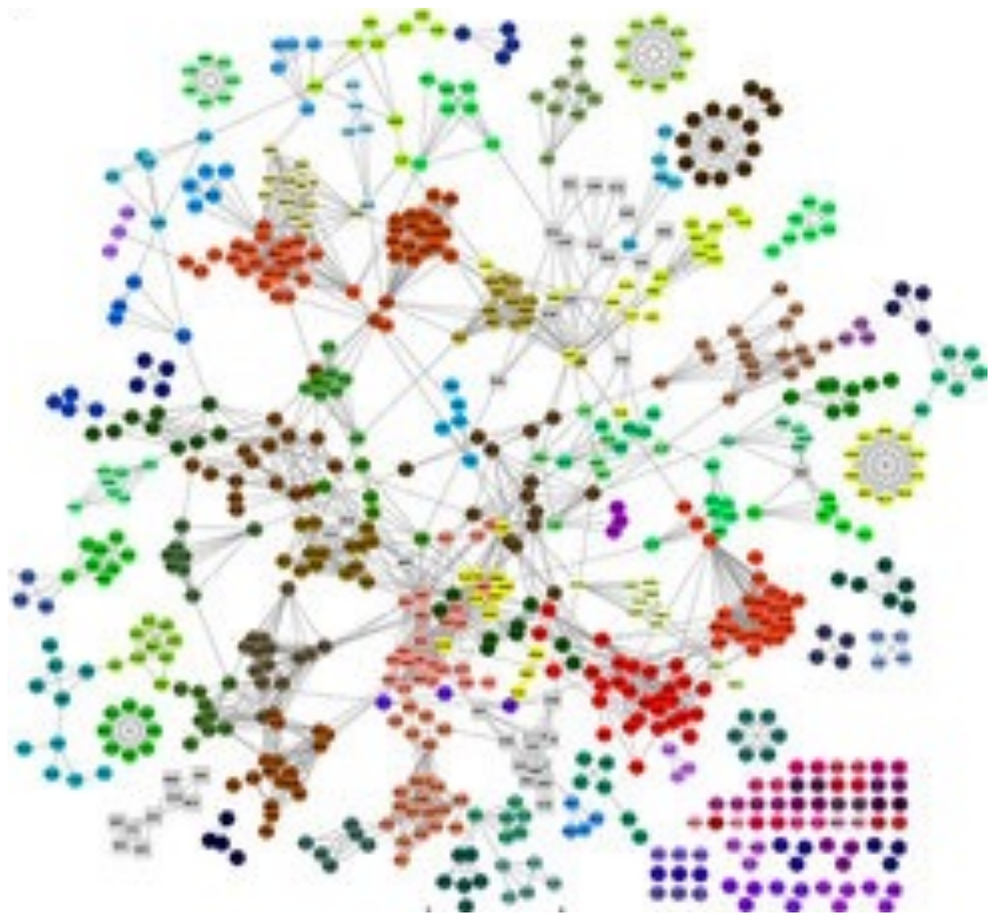
- Small degree
- Average path length proportional to *ln(ln(#nodes))*
- Transitivity high than Small-World and Scale Free
- Betweenness lower than Small-World and Scale Free



Degree = 4.0
Transitivity = 0.04
Betweenness = 290.4

Community or Cluster

One of the main feature of the biological network is the presence of communities or clusters.



Gaiter, Scientific Reports 2015

Cluster are important to detect similarity between nodes (genes, diseases, etc) in the same cluster.

Network Robustness

Robustness, the ability to withstand failures and perturbations. It is a critical attribute of many complex systems including biological networks.

Robustness is tested removing nodes and checking if connections between the remaining nodes are conserved. This is possible because may exist alternative paths between two distinct nodes.

Biological networks persists despite the environmental noise, mutations etc.

Telecommunication networks resit to the attach of hackers and hardware failure

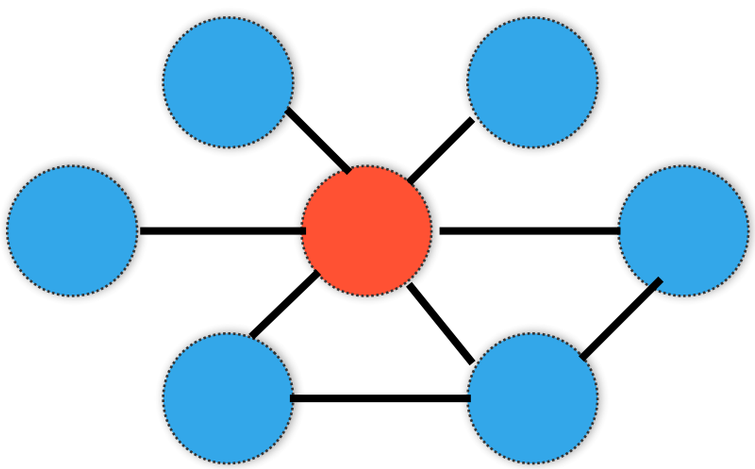
Network Attack

For random networks the effect of removing a single node is on average the same.

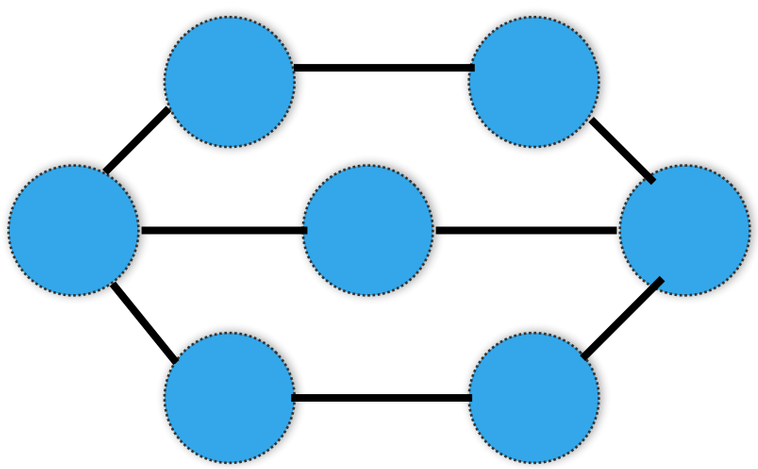
Biological networks are characterized by a small fraction of nodes with high degree (hubs)

An attack that aims to a hub has strong effect on the connectivity of the network.

In normal situation we assume that attacks are random. Thus, on average, an attach should have smaller effect on Biological Network.



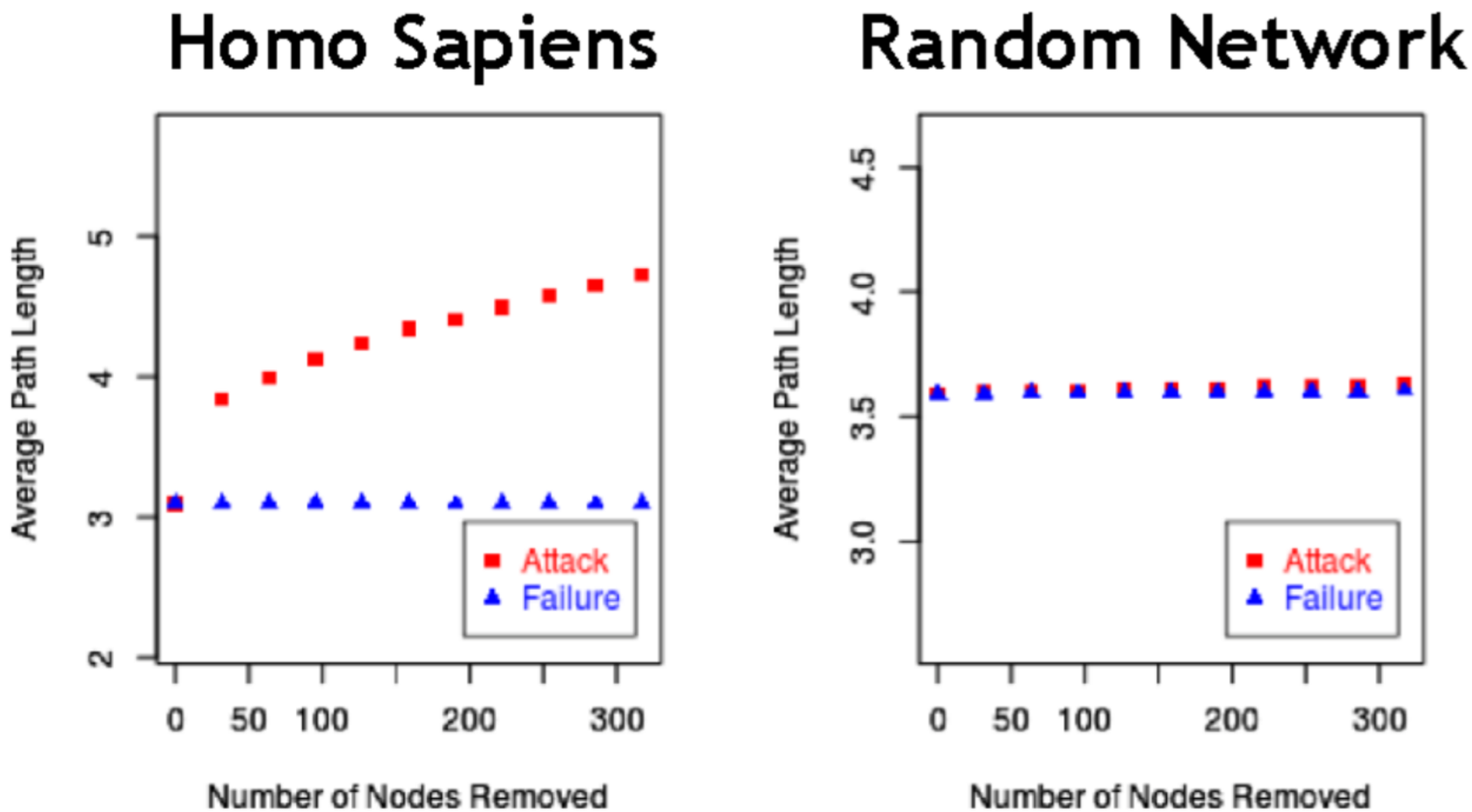
Biological network



Random network

Multiple attacks

Removing small number of nodes has less impact on biological network with respect to random network. Stronger effect is shown when the number of affected nodes increases.



Python NetworkX

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

```
>>> import networkx as nx
>>> G = nx.Graph()


>>> G.add_node(1)
>>> G.add_nodes_from([2, 3]) # add list of nodes


>>> G.add_edge(1, 2)
>>> G.add_edges_from([(1, 2), (1, 3)]) # add list of edges


>>> G.number_of_nodes()
3
>>> G.number_of_edges()
2
```

Könisberg Graph

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

```
>>> import networkx as nx
>>> M = nx.MultiGraph()

>>> M.add_edges_from([(1, 2, {“name”:“A”}),
...                   (1, 2, {“name”:“B”}), (1, 3, {“name”:“C”}),
...                   (1, 3, {“name”:“D”}), (1, 4, {“name”:“E”}),
...                   (3, 4, {“name”:“F”}), (2, 4, {“name”:“G”})])

>>> M = M.degree(1)
```

Network generators

Networkx has function that generate standard network types

```
>>> import networkx as nx
>>> import matplotlib as plt
```

```
>>> er = nx.erdos_renyi_graph(100, 0.15)
>>> ws = nx.watts_strogatz_graph(30, 3, 0.1)
>>> ba = nx.barabasi_albert_graph(100, 5)
```

```
>>> nx.draw(nx)
>>> plt.show()
```

Exercise

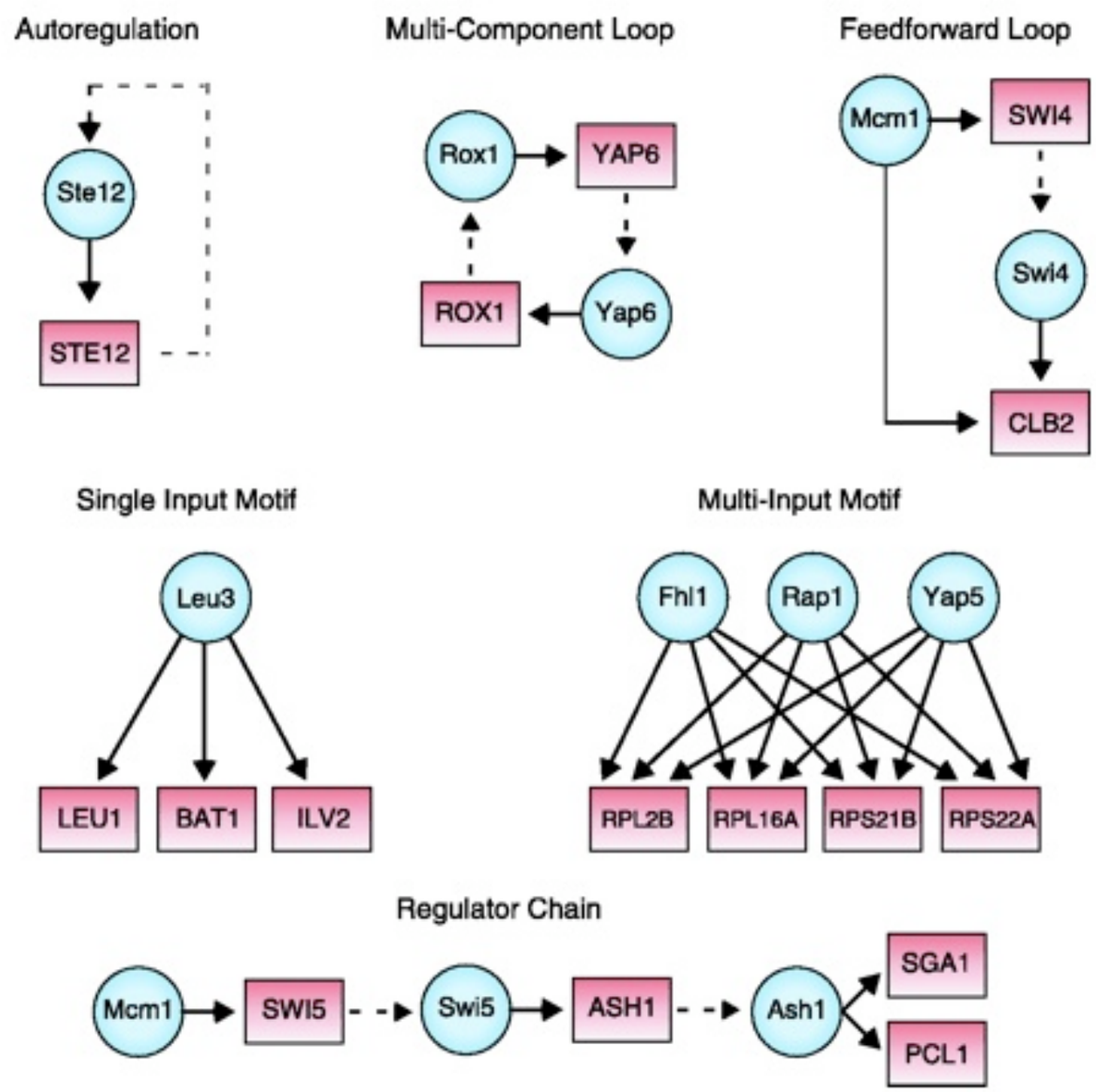
Generate the three types of network (random,”small world” and “scale free”) and calculate the distribution of the degree, betweenness and clustering.

From BioGRID download the Yeast interactome and analyze it with networkx importing only a list of unique interactions form the following files:
http://biocomp.unibo.it/emidio/tmp/biogrid-yeast-mitab.txt.gz
http://biocomp.unibo.it/emidio/tmp/biogrid-human-mitab.txt.gz

- How many components are present?
- What is the gene with highest degree?
- What is the the average values of degrees, betweenness and clustering?

Network Motifs

Network analysis is important for detecting network motifs, which are recurrent and statistically significant sub-graphs or patterns.



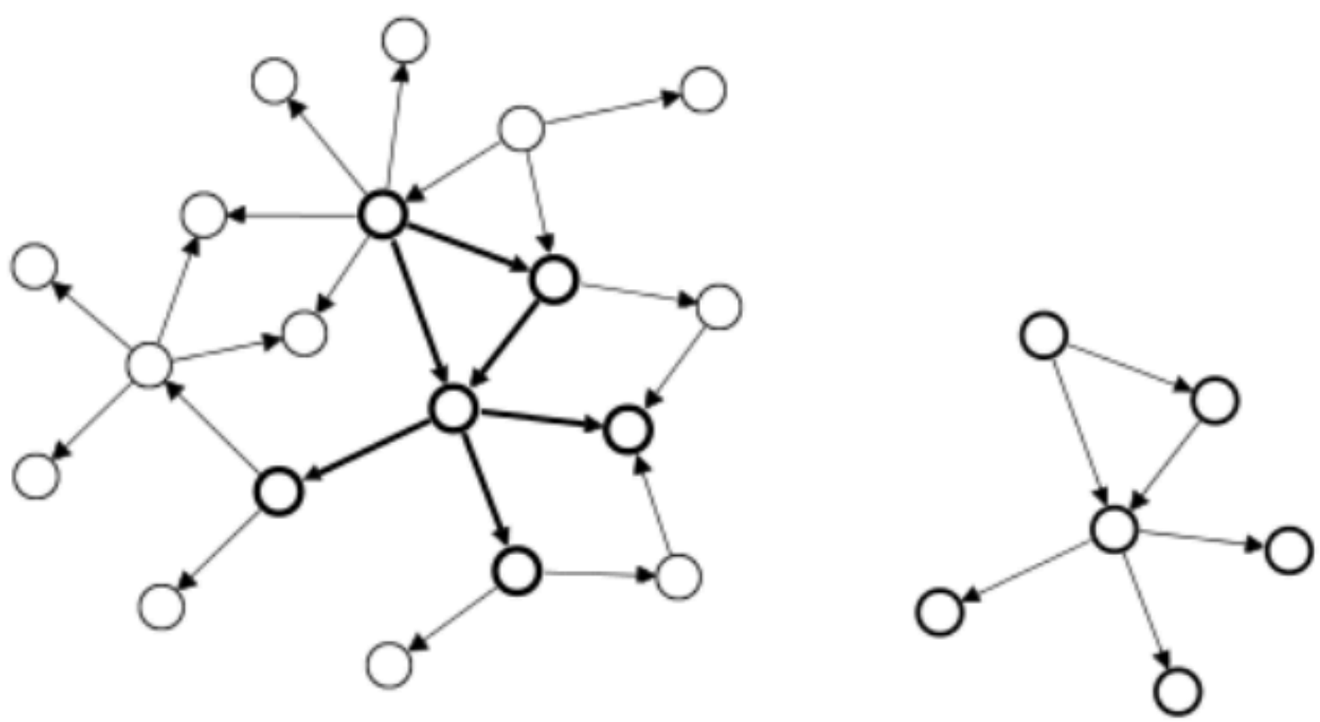
http://compbio.pbworks.com/

Motif Matching

A match of a motif G’ in the target graph G = (V, E) is a subgraph G” = (V”, E”) which is isomorphic to motif G’

Two graphs G’ and G” are isomorphic if there is a bijective mapping between the edge and vertex identities

i.e. G‘ is transformed to G” by changing the vertex and edge identities



Problem Complexity

The complexity of graph isomorphism is in the 'grey area' of complexity:

- It belongs to NP class of problems (problems where solution is easy to verify once found)

- It is not known if graph isomorphism belongs to P class of problems (problems that can be solved efficiently)

- It is not known if graph isomorphism is NP-complete (problems that are believed to be hard ot solve but easy to verify)

- Subgraph isomorphism, checking if a subgraph G" that is isomorphic to given graph G' exists in a larger graph G, is known to be NP-complete

- No hope for really fast algorithms for finding motifs.

Statistical Significance

A motif is a statistically overrepresented pattern of local interactions in the network

- Overrepresentation = occurring more frequently than expected by chance

- The motif has emerged several times therefore it has been conserved in the evolution of the network

- The rationale is that overrepresentation may denote possible conservation of the function

Significance tests

The statistical significance can be tested calculating the z-score of the presence of the motif on a set of randomly generated graphs obtained with

- Generation of random networks with the Erdos-Renyi algorithm

- Random shuffling of the edges

Detection of Motifs

Networkx allow to select a subgraph of a the whole graph and verify if two graphs are isomorphic

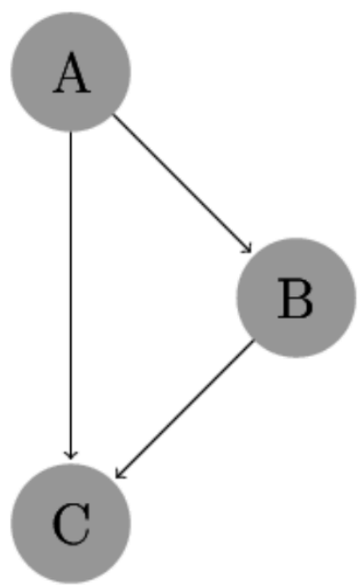
```
>>> g = nx.Graph().add_nodes_from([(1,2),(1,3)])

>>> mot = nx.Graph().add_nodes_from([(“A”,”B”)])

>>> g1 = g.subgraph([1,2])

>>> nx.is_isomorphgc(g1,mot)
```

Exercise

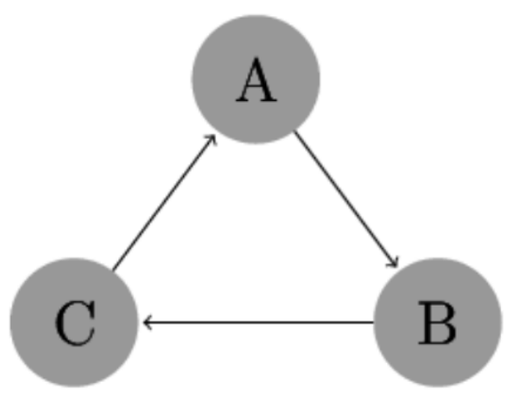
Given the Feed Forward Loop (FFL) and 3-Cycle write the code to detect the motif in the graph with 6 nodes and 8 edges.

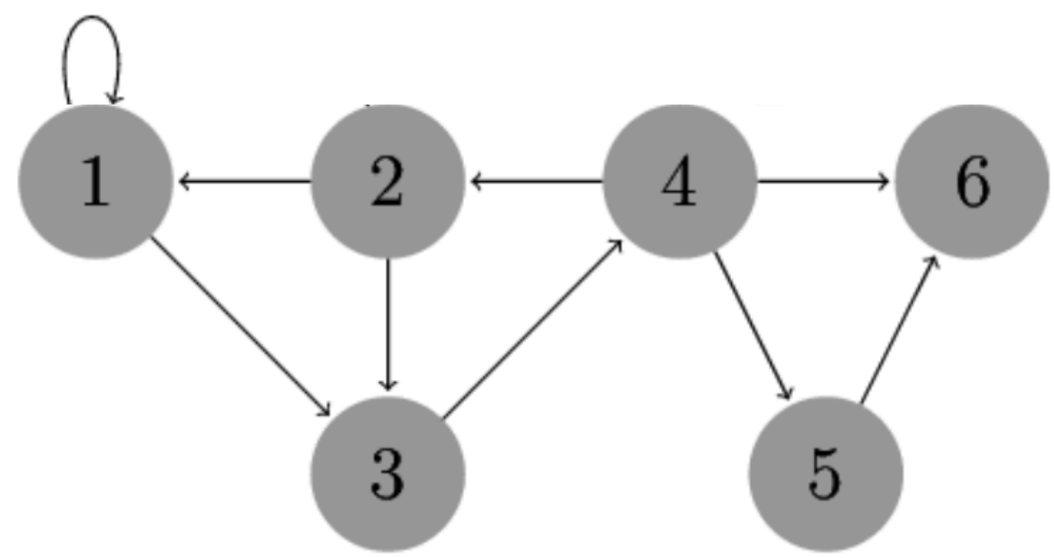
Calculate occurrence on random network and the z-score.

Feed Forward Loop



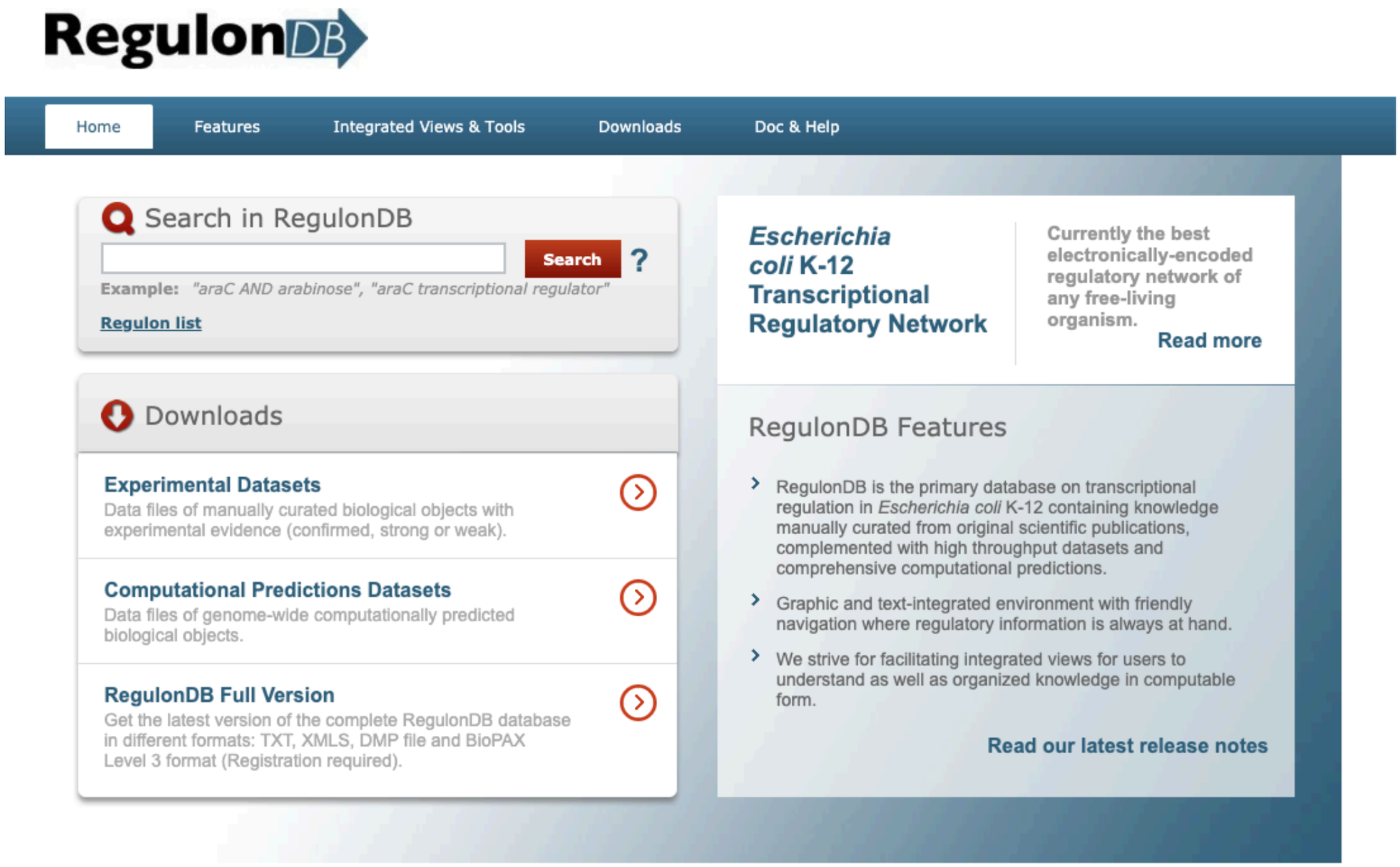
3-Cycle





RegulonDB

Database of Escherichia coli K-12 Transcriptional Regulatory Network



http://regulondb.ccg.unam.mx/

Regulation Data

The regulation data includes information about the transcription factors (TF) that activate or repress the expression of the genes with associated supporting evidences.

```
# Release: 10.6.2 Date: 10-04-2019
# ____________________________________________________#
# Columns:
#  (1) Transcription Factor (TF) name
#  (2) Gene regulated by the TF (regulated gene)
#  (3) Regulatory effect of the TF on the regulated gene (+ activator, - repressor, +- dual, ? unknown)
#  (4) Evidence that supports the existence of the regulatory interaction
#
AcrR    acrA    -       [BCE, BPP, GEA, HIBSCS] Strong
AcrR    acrB    -       [BCE, BPP, GEA, HIBSCS] Weak
AcrR    acrR    -       [AIBSCS, BCE, BPP, GEA, HIBSCS] Weak
AcrR    flhC    -       [GEA, HIBSCS]   Weak
AcrR    flhD    -       [GEA, HIBSCS]   Weak
AcrR    marA    -       [BPP, GEA, HIBSCS]      Strong
AcrR    marB    -       [BPP, GEA, HIBSCS]      Strong
AcrR    marR    -       [BPP, GEA, HIBSCS]      Strong
AcrR    micF    -       [AIBSCS]        Weak
AcrR    soxR    -       [BPP, GEA, HIBSCS]      Strong
```

http://regulondb.ccg.unam.mx/menu/download/datasets/files/network_tf_gene.txt

Nodes ad Edges

With networkx we can assign attributes to nodes and edges

```
>>> G=nx.DiGraph()
>>> G.add_node(1, color='blue')
>>> G.add_node(2, color='red')
>>> G.add_edge(1, 2, sign='+')
>>> G.node[1]
>>> G.edge[1][2]
```

Matches Node and Edges

Matches can be performed based on node and edges attributes

```
>>> import networkx.algorithms.isomorphism as iso

>>> em=so.categorical_edge_match(‘sign'=‘+’)

>>> nm=iso.categorical_node_match(‘color'=‘red’)

>>> nx.is_isomorphic(G1,G2,edge_match=em, node_match=nm)
```

Exercise

Write a program to analyze the RegulonDB network considering only data with strong supporting information.

- Find the TF that regulates more genes (activation and suppression)
- Find the gene that is regulated by more TFs
- Match a graph that contains a TF activating three genes.