

# NETWORK MODELING, VISUALIZATION AND ANALYSIS II

The igraph library for R and Python

**Master en Bioinformática**

Universitat de València (UV)

October 10<sup>th</sup>, 2013

**Marta Bleda Latorre**

PhD at the Computational Genomics Institute

Centro de Investigación Príncipe Felipe (CIPF)

Valencia, Spain

# Overview

---

- The igraph library
- Installation in R
- igraph basics
  - Exercise
- Network models and robustness
  - Exercise
- igraph for Python
- Homework :)



# The igraph library

- For complex network analysis
- Lots of graph algorithms implemented
- Core functionality is implemented as a C library
- High level interfaces for **R** and **Python**
- Open source
- <http://igraph.sourceforge.net/>

The igraph library

[Home](#) [News](#) [Download](#) [Documentation](#) [Wiki](#) [Screenshots](#) [Mailing lists](#) [Bugs](#) [License](#)

### Introduction

**igraph** is a free software package for creating and manipulating undirected and directed graphs. It includes implementations for classic graph theory problems like minimum spanning trees and network flow, and also implements algorithms for some recent network analysis methods, like community structure search.

[Read more »](#)

### Features

**igraph** contains functions for generating regular and random graphs, manipulating graphs, assigning attributes to vertices and edges. It can calculate various structural properties, graph isomorphism, includes heuristics for community structure detection, supports many file formats. The R and Python interfaces support visualization.

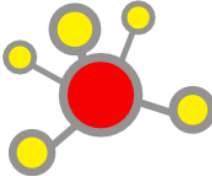
[Read more »](#)

### Requirements

igraph runs on most modern machines and operating systems, and it is tested on MS Windows, Mac OSX and various Linux versions.

The software you need for installing **igraph** depends on whether you want to use the C library, the R package or the Python extension; and may vary depending on your platform.

[Read more »](#)



Latest version: **0.6.5**  
[Release notes](#)

# What do we need?

## R & RStudio

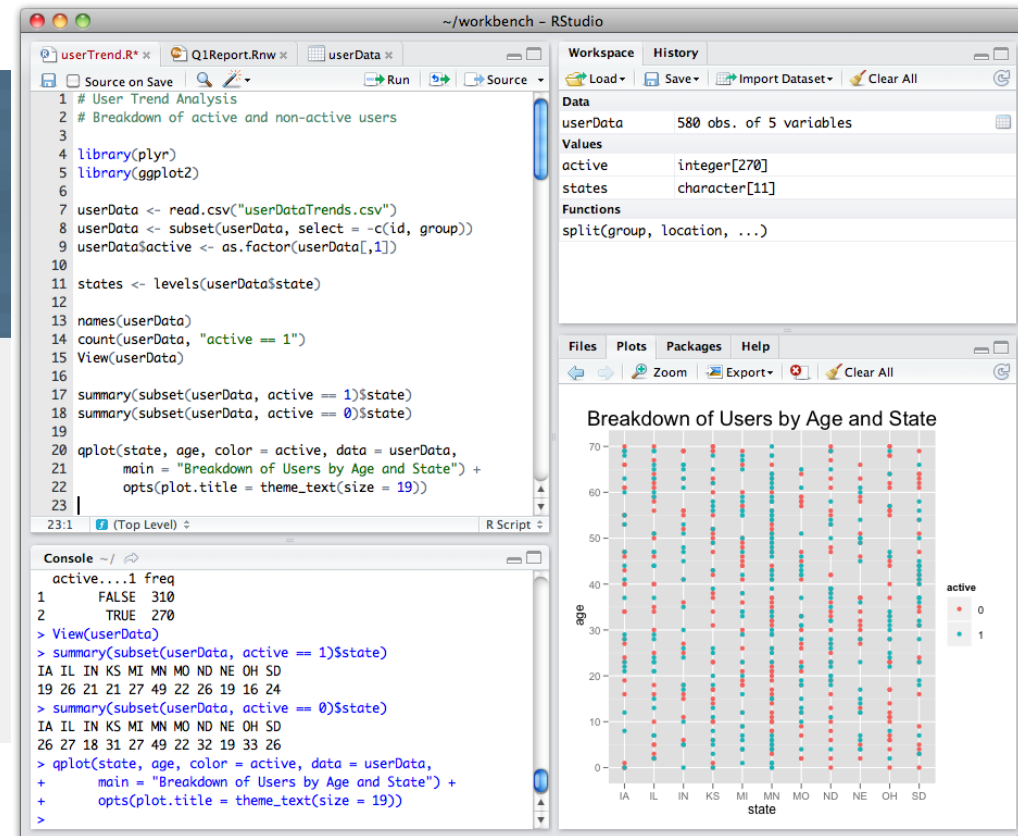
- If you don't have R installed just type (in Ubuntu shell):

```
$ sudo apt-get update  
$ sudo apt-get install r-base
```

- And now Rstudio...



<http://www.rstudio.com/>



# igraph installation in R

---

- Very easy!
- Enter the R shell and type:

```
> install.packages("igraph")
```

- Load it!

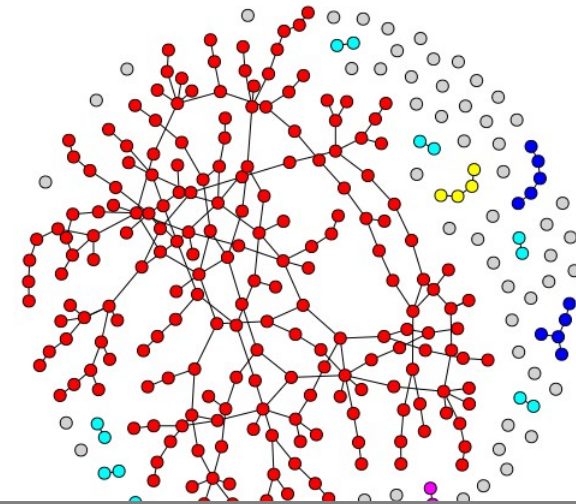
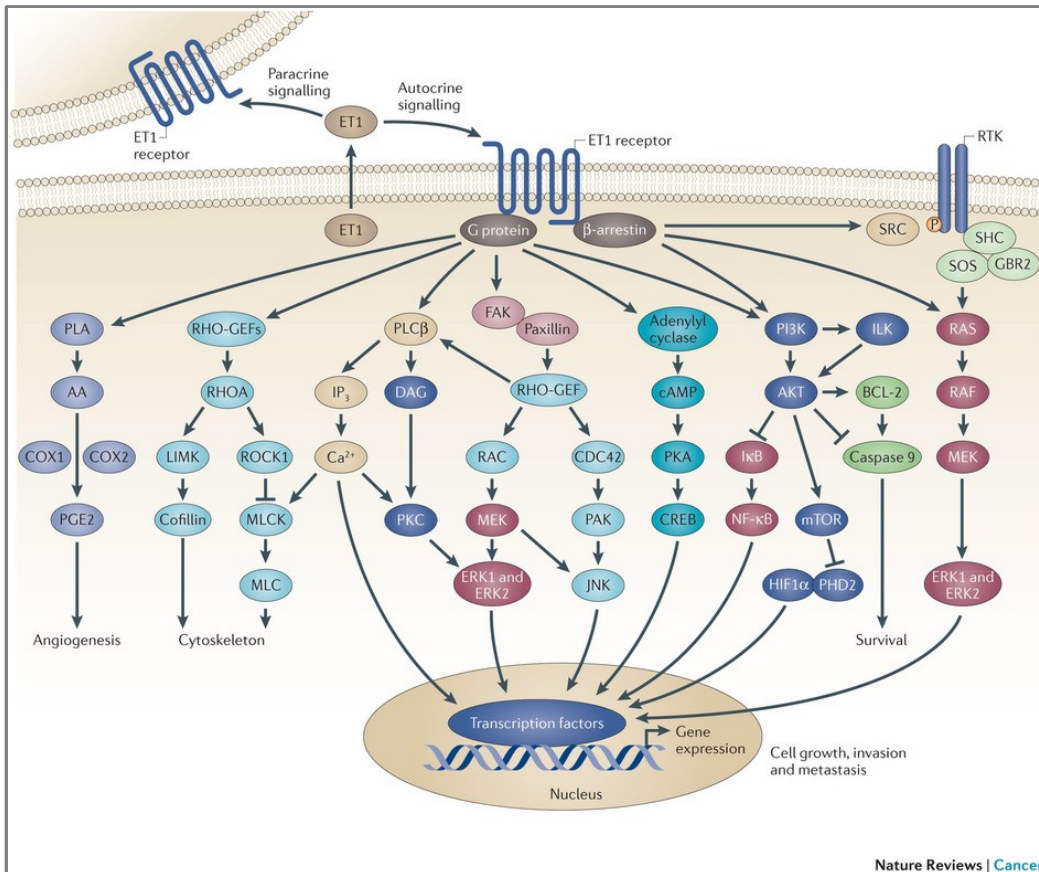
```
> library(igraph)
```

# The igraph library

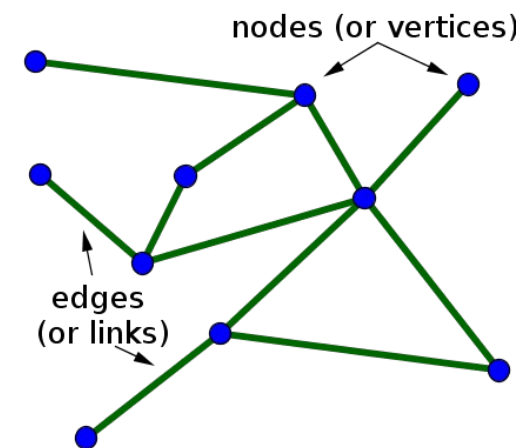
## Basic concepts

**Network** (informal concept)

**Graph** (formal mathematical object)



vertices = {A, B, C, D, E}  
edges = ({A,B}, {A,C}, {B,C}, {C,E})



# The igraph library in R

## Creating a graph

---

- Vertices and edges have numerical vertex ids in igraph. Vertex ids are always consecutive and they are **numbered from 1** (from igraph 0.6.5 onwards)
- To create a graph whose nodes are letters we have to ***translate*** vertex names to ids:

$V = \{A, B, C, D, E\}$

$E = ((A, B), (A, C), (B, C), (C, E))$

$A = 1, B = 2, C = 3, D = 4, E = 5$

```
> g <- graph( c(1,2, 1,3, 2,3, 3,5), n=5 )
```

- By default, igraph creates a **directed** graph, but we can change it:

```
> g <- graph( c(1,2, 1,3, 2,3, 3,5), n=5, directed=FALSE )
```

- Let's play a little bit around...

Open the R script **igraph\_basic\_concepts.R**

# The igraph library in R

## Accessing graph information

---

Description	Function
Vertex and edge counts	<code>vcount(g)</code>
	<code>ecount(g)</code>
Vertex and edge lists	<code>V(g)</code>
	<code>E(g)</code>
Vertex and edge attributes	<code>V(g)\$name</code>
	<code>E(g)\$weight</code>
Vertex and edge attribute lists	<code>list.vertex.attributes(g)</code>
	<code>list.edge.attributes(g)</code>



# The igraph library in R

## Accessing topological information

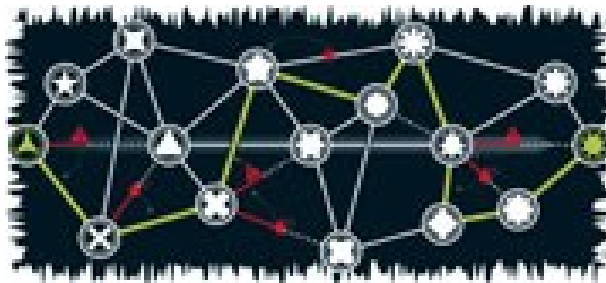
---

Description	Function
Node degree	<code>degree(g)</code>
Degree distribution	<code>degree.distribution(g)</code>
Betweenness	<code>betweenness(g)</code>
Closeness	<code>closeness(g)</code>
Clustering coefficient	<code>transitivity(g)</code>
Shortest paths	<code>shortest.paths(g)</code>
	<code>get.shortest.paths(g)</code>
Diameter	<code>diameter(g)</code>
	<code>get.diameter(g)</code>

# Error and attack tolerance of complex networks

---

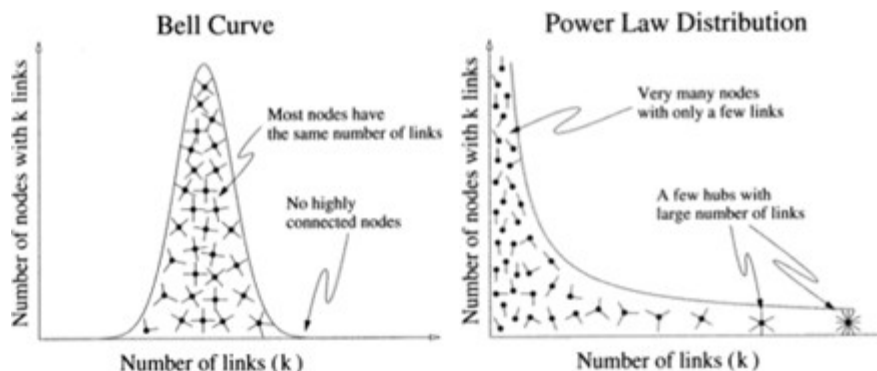
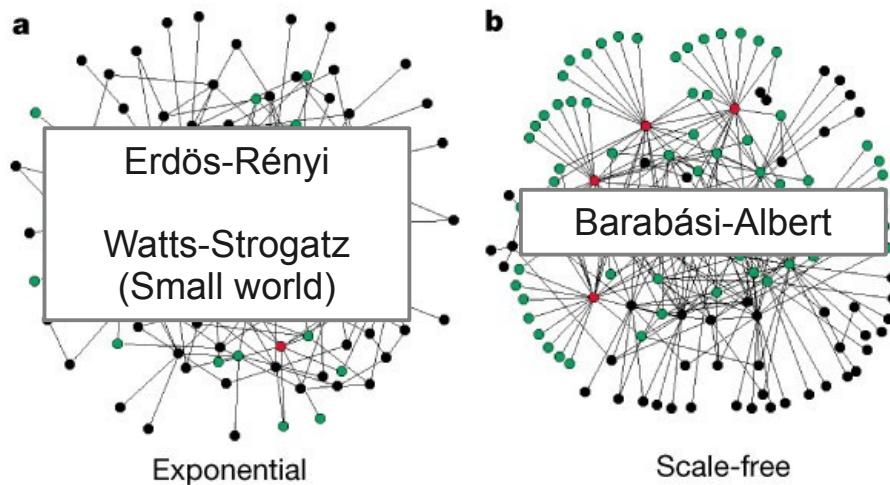
- Components regularly **malfunction**
- However, this failure rarely perturbs the ability of the network → Many complex systems display a surprising degree of **tolerance**
- Stability attributed to the **redundant wiring** of the functional web
- **Robustness**: There must be at least two alternative paths between any two points on the network, so that removal of any one component leaves the network fully operational.
- But... what happens with metabolic networks?
- However, extremely **vulnerable** to the attack of a few nodes



# Network models

## Types of networks

Complex networks can be divided into **two major classes** depending on the degree distribution



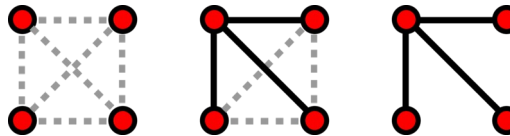
- 130 nodes and 215 links
  - **RED**: 5 nodes with the highest number of links
  - **GREEN**: The first neighbours
  - Exponential, only 27% of the nodes are reached
  - Scale-free, > 60% of the nodes are reached
- 
- Mechanism:
    - Growth
    - Preferential attachment

# Network models

## Types of networks

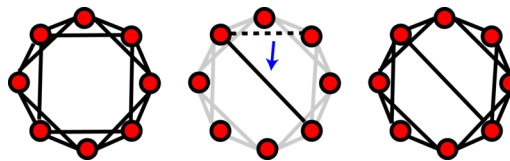
- **Random Networks (Erdős-Rényi (ER))**

Every node is connected to a randomly selected number of other nodes with **equal probability**. Two parameters needed: the number of nodes and the probability that a link should be formed between any two nodes.



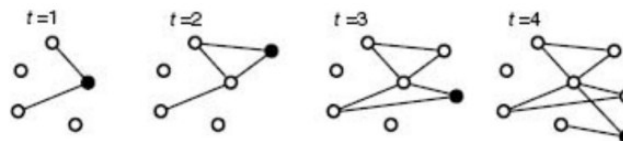
- **Small-World Networks (Watts-Strogatz (WS)):**

Produces graphs with small-world properties, including short average path lengths and high clustering.



- **Scale-free Networks (Barabási-Albert (BA))**

A scale free network is a network that its degree distribution obeys a power law:  $P(k) \sim k^{-\gamma}$ . Model used to demonstrate a preferential attachment or a "**rich-get-richer**" effect.



# Vulnerability and robustness

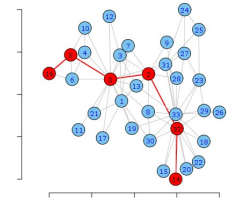
---

- A **perturbation** in a network can be defined as a set of elementary changes:
  - Failure of a node
  - Failure of a link
  - Instability of a network due to removal of a stabilizing link
- 2 types of attacks
  - **Random attack**: Removal of nodes randomly (% of network)
  - **Targeted attack**: Removal of nodes with particular attributes values (highest degree nodes)
- Article:  
*Error and attack tolerance of complex networks*. Albert R, Jeong H, Barabasi AL. *Nature*. 2000 Jul 27;406(6794):378-82.

# Vulnerability and robustness

## Network diameter

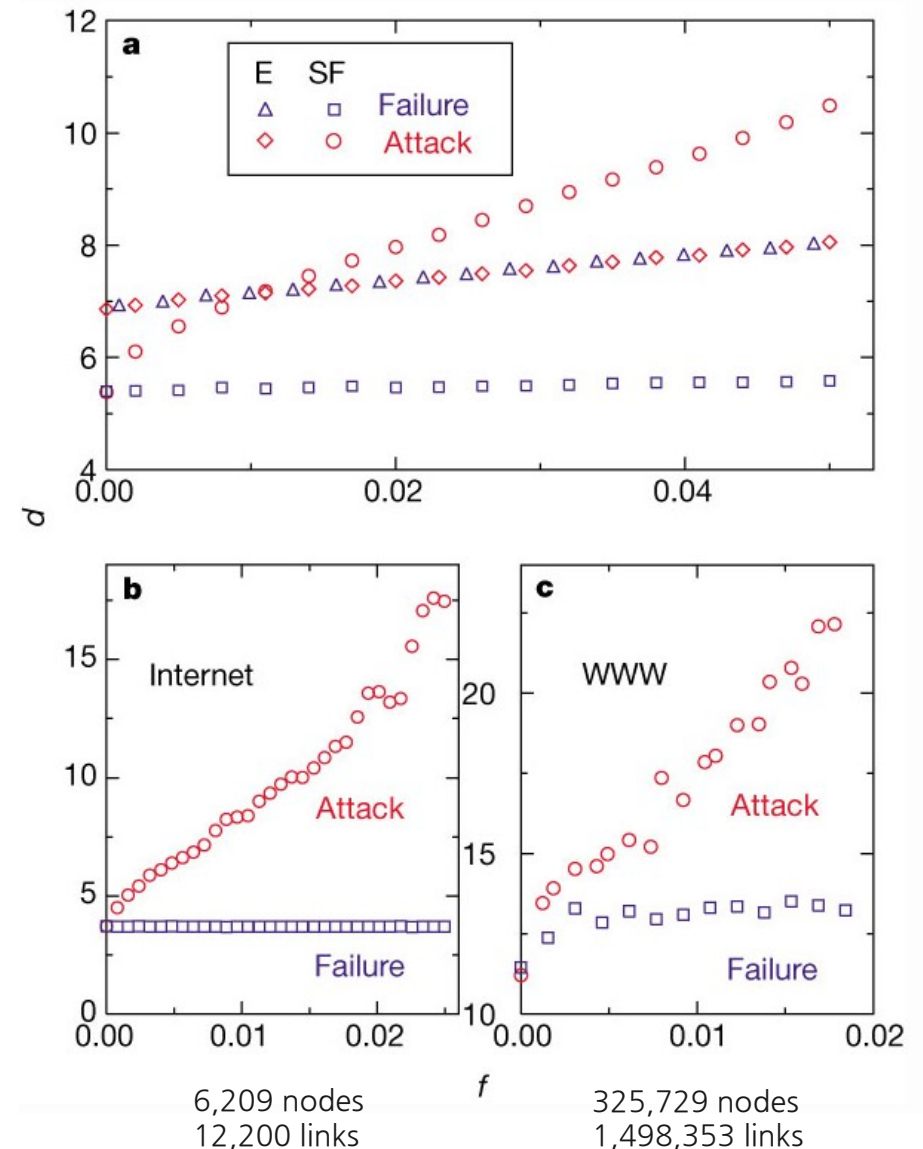
Diameter of the Zadiary Karate Club network



created by graph 0.4

- **2 networks:** a random network (ER) and a scale-free network (BA)
  - 10,000 nodes and 20,000 links
- $d$  = diameter,  $f$  = % nodes removed randomly
- Attack over the **most connected nodes**

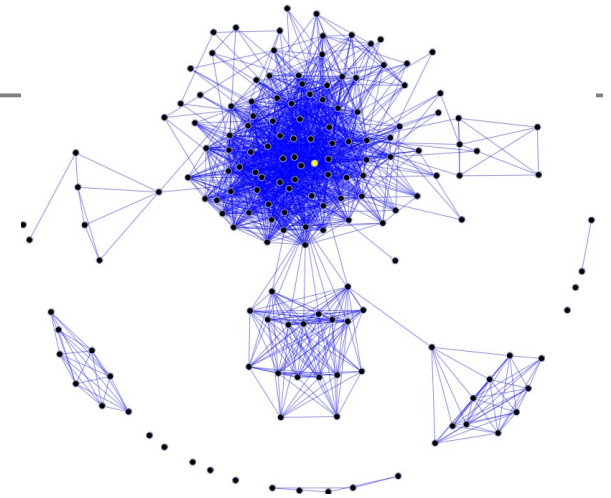
	Random network	Scale-free network
Failure	<p>Diameter <b>increases monotonically</b> with <math>f</math>.</p> <p>All nodes have ~ same number of links → <b>contribute equally</b>.</p>	<p>Diameter remains <b>unchanged</b>.</p> <p>Lots of nodes with low connectivity → Removal of “small nodes” does <b>not alter</b> the path structure.</p>
Attack	<p>Due to the <b>homogeneity</b> of the network there is <b>no substantial difference</b> with failure attack.</p>	<p>Drastically different behavior → <b>diameter increases rapidly</b>.</p> <p>Vulnerability due to <b>inhomogeneity</b> → connectivity maintained by a few highly connected nodes.</p>



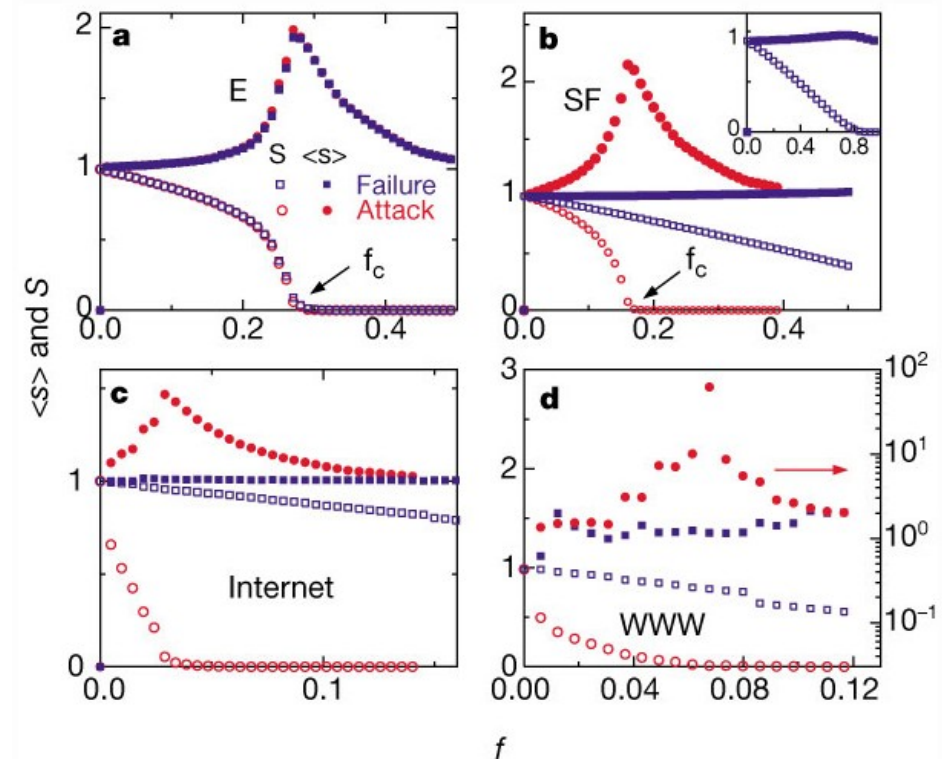
# Vulnerability and robustness

## Network fragmentation

- When nodes are removed, clusters of **nodes** whose links to the system disappear may be cut off (**fragmented**) from the **main cluster**
- $S$  = size (number of nodes) of the largest component (cluster)
- $\langle s \rangle$  = average size of the isolated components, except the largest one



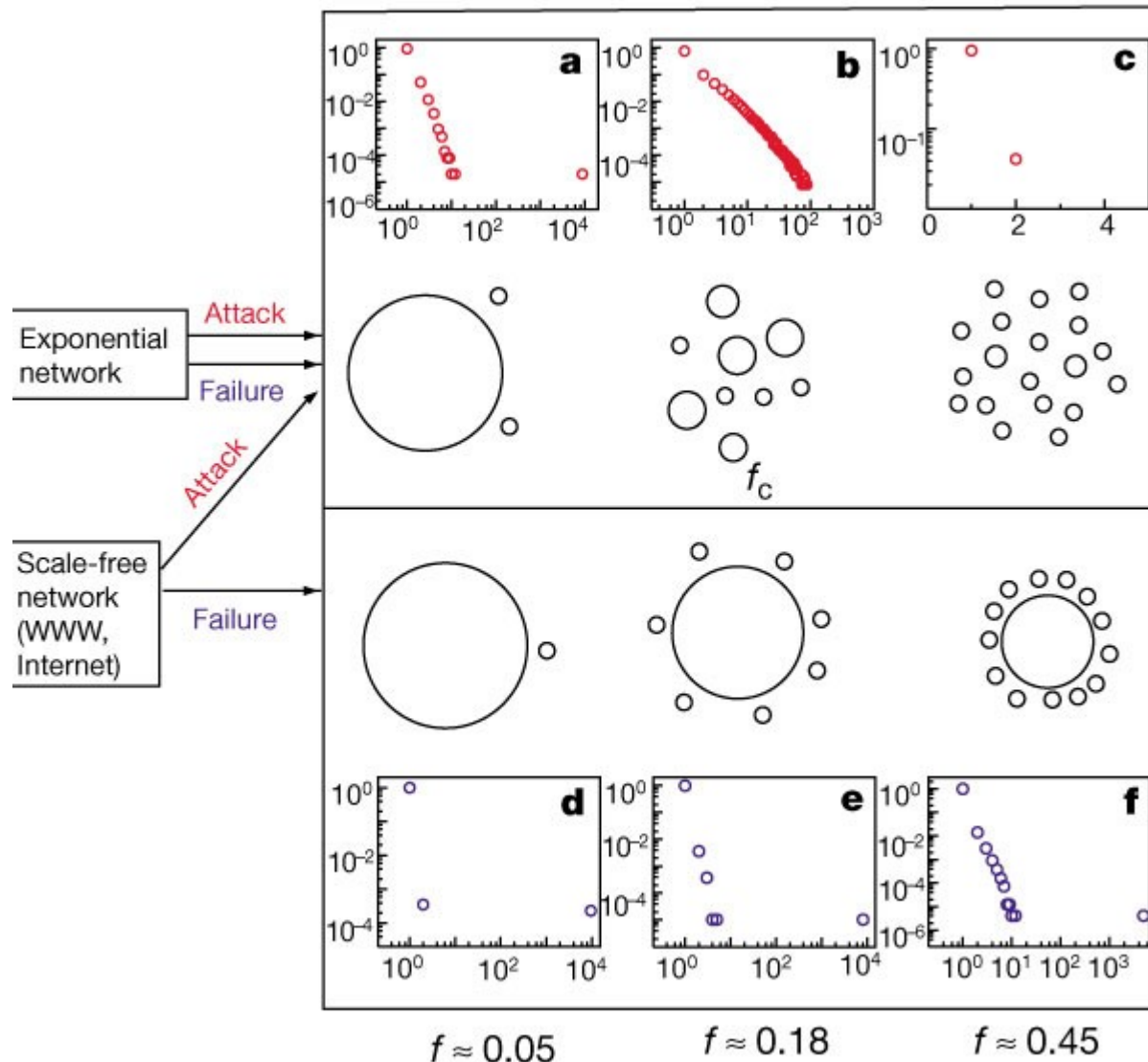
	Random network	Scale-free network
Failure	<p>No difference between failure and attack.</p> <p><math>S</math> decreases rapidly until the main cluster is completely fragmented <math>\rightarrow S \sim 0</math>.</p>	<p><math>S</math> decreases slowly without observed threshold.</p> <p><math>\langle s \rangle \sim 1</math>, network deflated by nodes breaking off one by one.</p> <p>Largest component stays together for very high <math>f \rightarrow</math> <b>stability</b></p>
Attack	<p><math>\langle s \rangle</math> increases rapidly until <math>\langle s \rangle = 2</math>, after which it decreases to <math>\langle s \rangle = 1</math>.</p>	<p>Similar to a random network but falling apart at a smaller value of <math>f</math>.</p>





# Vulnerability and robustness

## Conclusions



- **Random graphs** are more **sensitive** than scale-free when nodes are removed randomly.
- **Scale-free networks** display a surprising high degree of **tolerance** against random failure when compared to random networks.
- Under a **targeted attack**, scale-free networks increase their diameter rapidly and break into many **isolated fragments**.



# Homework

## Simulate and attack

---

### 1. Simulate 3 networks

- Erdős-Rényi ( $n=1000$ ,  $p.or.m=2/1000$ ,  $type="gnp"$ )
  1. Watts-Strogatz ( $dim=1$ ,  $size=1000$ ,  $nei=1$ ,  $p=0.5$ )
  2. Barabási-Albert ( $n=1000$ )

### 2. For these 3 networks, perform a random and a targeted attack damaging progressively the 0%, 0.1%, 0.2%, 0.3%, 0.4%, 0.5%, 0.6%, 0.7%, 0.8%, 0.9%, 1%, 1.5%, 2% and 2.5% of the nodes.

### 3. Study the variability of the following parameters:

1. Average degree
2. Diameter
3. Size of the giant component
4. Betweenness

### 4. Summarize this information in a plot and comment your results briefly.

THANK YOU.