TFY4235/FY8904: Computational Physics (spring 2019)
Assignment 1 – Percolation

Arnau Sala

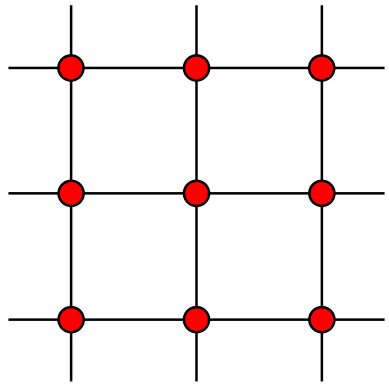
Department of physics, NTNU

What is Percolation?

What is percolation?

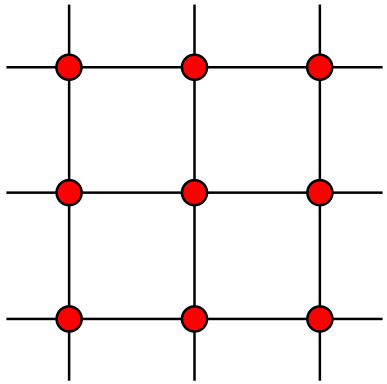
Percolation is the study of the evolution of clusters in a network.

Consider a system composed of N elements that interact between them. Place them in a fixed position (as in a lattice) and make this interaction a short-range interaction.



Here is your network. The sites (red dots) are connected to their nearest neighbors via some links or bonds (black lines).

Clusters



These bonds can be though of as the spin-spin interaction in an antiferromagnetic material (the nodes would be the spins) or the physical links connecting the principal data routers of the Internet.

A cluster is formed when two or more neighboring spins point in the same direction (the probability of activating a bond is a function of the temperature) or by creating a group of data routers connected with links.

Bond percolation

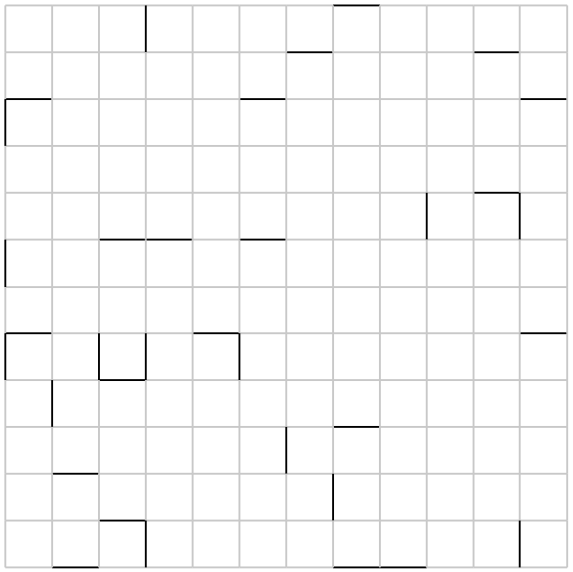
In this assignment we will focus on bond percolation: we have a set of sites and activate the bonds one by one randomly.
Let's see an example

This is a square lattice. There are 13×13 nodes (vertices) connected to their nearest neighbors by some bonds (gray lines). At the beginning these bonds are not activated and: there are no clusters, the sites are isolated.

Bond percolation

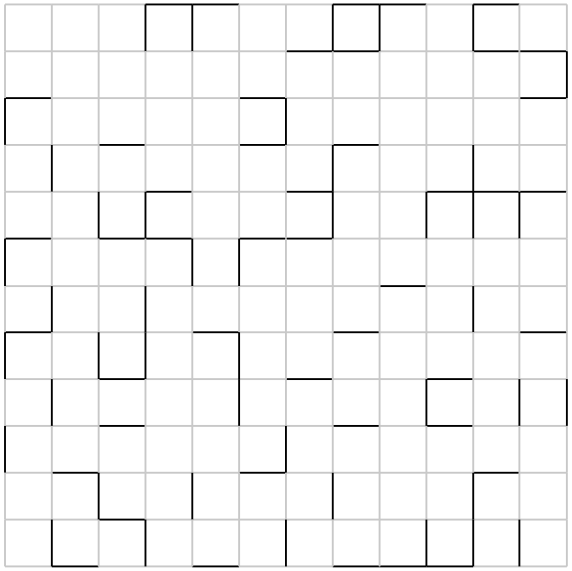
Now let's start activating the bonds one by one randomly: choose a bond, just any, and activate it (or not) with some probability p.

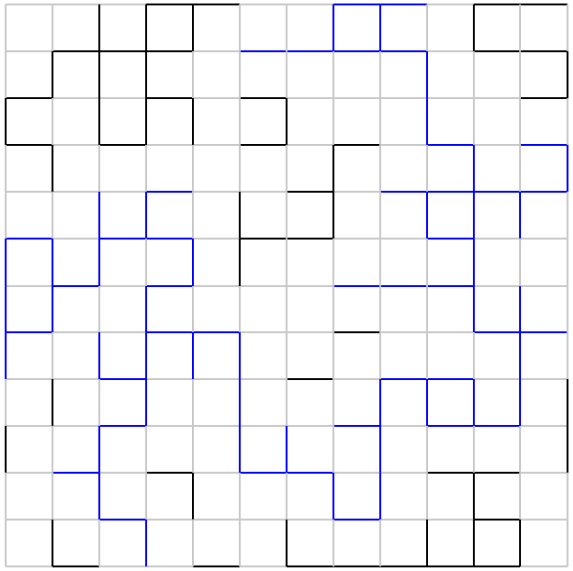
If p is small we will have some small clusters...



Bond percolation

... but as we increase p these will soon start to grow





until one of the clusters (blue) gets big enough to span across the network. When this happens we say that the system has percolated.

Percolation threshold

This probability p is the control parameter of the system. By changing p we can go from a phase where the system is composed of many small clusters to a phase where one big cluster dominates over the others.

What is the number of bonds in the Internet network of data routers that I can cut (deactivate) before Internet stops working? The answer to this question is given by p_c: the probability at which the system percolates.

BTW, for the Internet the probability p_c is very small, meaning that you would need to cut many bonds.

Phase transitions

The goal of percolation is to find the critical probability p_c (or percolation threshold) at which the phase transition occurs.

Phase transitions are also characterized by other parameters. Around the phase transition the size of the largest cluster P_∞, the average cluster size $\langle s \rangle$ and the correlation length ξ scale as

$$P_\infty \propto (p - p_c)^\beta,$$
$$\langle s \rangle \propto |p - p_c|^{-\gamma},$$
$$\xi \propto |p - p_c|^{-\nu}.$$

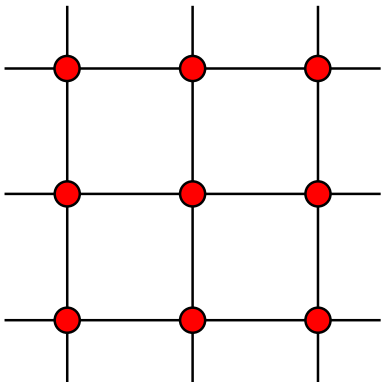
Your task will be to find p_c, β, γ and ν for different networks using the algorithm that we have described in the Assignment notes.

The networks

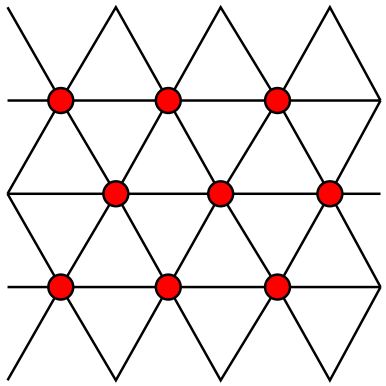
Lattices

We will start with something simple: lattices. These are periodic networks in which all the nodes are connected to exactly the same number of neighboring nodes.

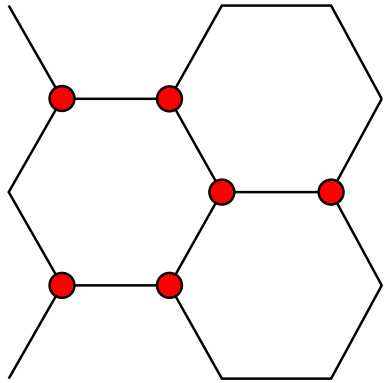
The lattices we will consider are:



1. Square lattice



2. Triangular lattice



3. Honeycomb lattice

Random networks

And after that we will switch to random networks, where the number of bonds k (this is also called *degree*) that link each node is given by a probability density function (PDF). We will consider two different random networks:

- An exponential random network, with a degree distribution given by the discrete PDF

$$P(k) = p(1-p)^k.$$

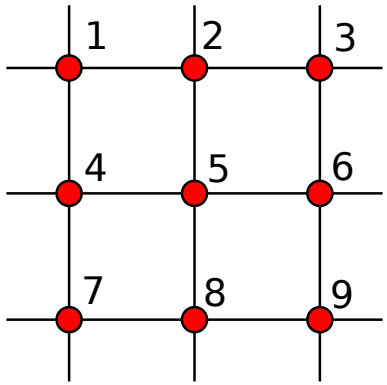
- A power-law random network, with a degree distribution given by the continuous PDF

$$P(k) = \frac{\gamma - 1}{k_0} \left(\frac{k}{k_0} \right)^{-\gamma}.$$

To generate these networks we will use the configuration model (more on the Assignment notes).

First task: Generate the network

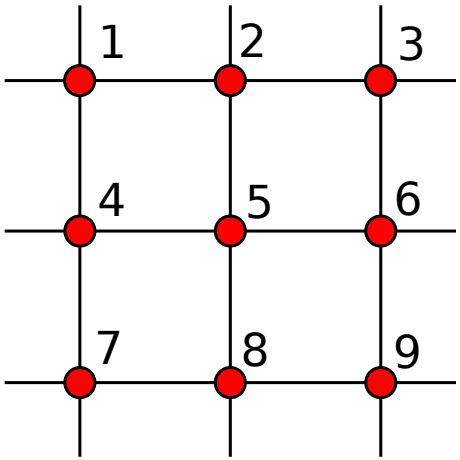
In the first task you are asked to create the network. Here what we want is a complete list of bonds.
Given a square lattice like this



We need a list that contains the bond that links sites 1 and 2, the bond that links sites 2 and 3, the bond that links sites 3 and 6, etc.

First task: Generate the network

For a 3×3 square lattice with periodic boundary conditions the list should look like this:



1	2
1	4
2	3
2	5
3	1
3	6
4	5
4	7
5	6
5	8
6	4
6	9
7	8
7	1
8	9
8	2
9	7
9	3

Note that by counting only the left and bottom bond of each site we count them all only once. Avoid repetitions!

Shuffle the list of bonds

This algorithm requires us to activate the bonds one by one randomly. This means that we have to take one random pair of sites of that list but being careful to not take the same pair twice.

To avoid repetitions we will shuffle the list and take each pair following the new order.

In the following slide you'll find the algorithm to shuffle the previous list.

Shuffle the list of bonds

Take a random number r (an integer) between 2 and 18 and swap the first and the r-th rows. For $r = 7$ this is

$$\begin{array}{cc} 1 & 2 \\ 1 & 4 \\ 2 & 3 \\ 2 & 5 \\ 3 & 1 \\ 3 & 6 \\ 4 & 5 \\ 4 & 7 \\ \vdots & \vdots \end{array} \longrightarrow \begin{array}{cc} 4 & 5 \\ 1 & 4 \\ 2 & 3 \\ 2 & 5 \\ 3 & 1 \\ 3 & 6 \\ 1 & 2 \\ 4 & 7 \\ \vdots & \vdots \end{array}$$

Next take a random number r between 3 and 18 and swap the second and the r-th rows. Repeat until you reach the end of the list.

Start activating bonds

At each iteration, after adding a bond you have to measure the giant component P_∞, the weighted average cluster size $\langle s \rangle$ and P_∞^2. You also have to keep track of the status of your network in an array `sites(i)` with dimension N, the number of sites of the network.

`sites(i)` contains the status of each site. If a site s is the root site of a cluster then `sites(s)` contains the number of sites of that cluster. Otherwise it contains the index of the root node.

To distinguish between indexes and sizes you can write the size with a negative integer and the index with a positive integer.

P_∞, $\langle s \rangle$ and P_∞^2 are also vectors with dimension M, the number of bonds (see the assignment notes for more information).

Start activating bonds

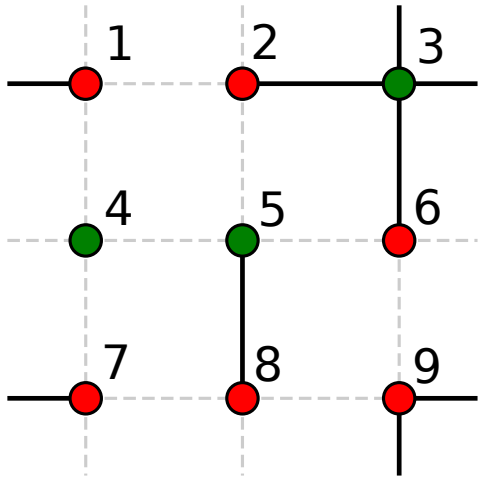
At the beginning, before activating any bond (0-th iteration), all the sites are isolated. They all are, thus, root nodes. In this case the variables you have to measure look like this (for a square lattice with $N = 9$):

$$\texttt{sites} = (-1, -1, -1, -1, -1, -1, -1, -1, -1)$$
$$P_\infty(0) = 0$$
$$\langle s \rangle = 1$$

Start activating bonds

And let's start activating bonds. After the 6-th iteration (we have activated 6 bonds) the lattice may look like this:

Here we have activated the bonds (2,3), (3,1), (3,6), (5,8), (9,7) and (9,3). The green sites are root sites and the red ones are just normal sites. At this iteration (the 6-th) we have:



$$\texttt{sites} = (3, 3, -6, -1, -2, 3, 9, 5, 3)$$
$$P_\infty(6) = 6/9$$
$$\langle s \rangle = 5/3.$$

$\langle s \rangle$ is the average of the squared cluster sizes, but without considering the largest cluster: $\langle s \rangle = (1^2 + 2^2)/3$

Recursive functions

Every time you add a bond you have to check the two sites that will now be linked. If they already belong to the same cluster no further actions are required, but if they do not belong to the same cluster you have to merge them. That is, in the array `sites(i)` the two sites that are now linked should point to the same root node.

You can do this using a recursive function. This is a function that calls itself:

- Given a site s find where this site is pointing to. That is, read `sites(s)`.
- If it points to a root node, then return the root node `r` = `sites(s)`.
- Otherwise call the function again with `sites(s)` as argument instead of s. That is, `sites(sites(s))`

Recursive functions

This is how the function should look like in python:

```
def findroot(s):
    if sites[s] < 0:
        return s
    else:
        sites[s] = findroot(sites[s])
        return sites[s]
```

And in Fortran:

```
recursive function findroot(s) result(r)
  if (sites(s) < 0) then
    r=s
  else
    r=findroot(sites(s))
    sites(s)=r
  end if
end function findroot
```

Take pictures

One of the tasks consists in taking *pictures* of the system after activating some bonds. This means that every now and then you have to make a list of all the sites that belong to the largest cluster (a list of all the sites that have the same root $\rightarrow$ use your recursive function here too).

Once you have this list you have to transform the label of each site into coordinates in a 2-dimensional plane. For a square lattice with lateral size L and sites numbered from 0 to $N-1$, a site s has coordinates:

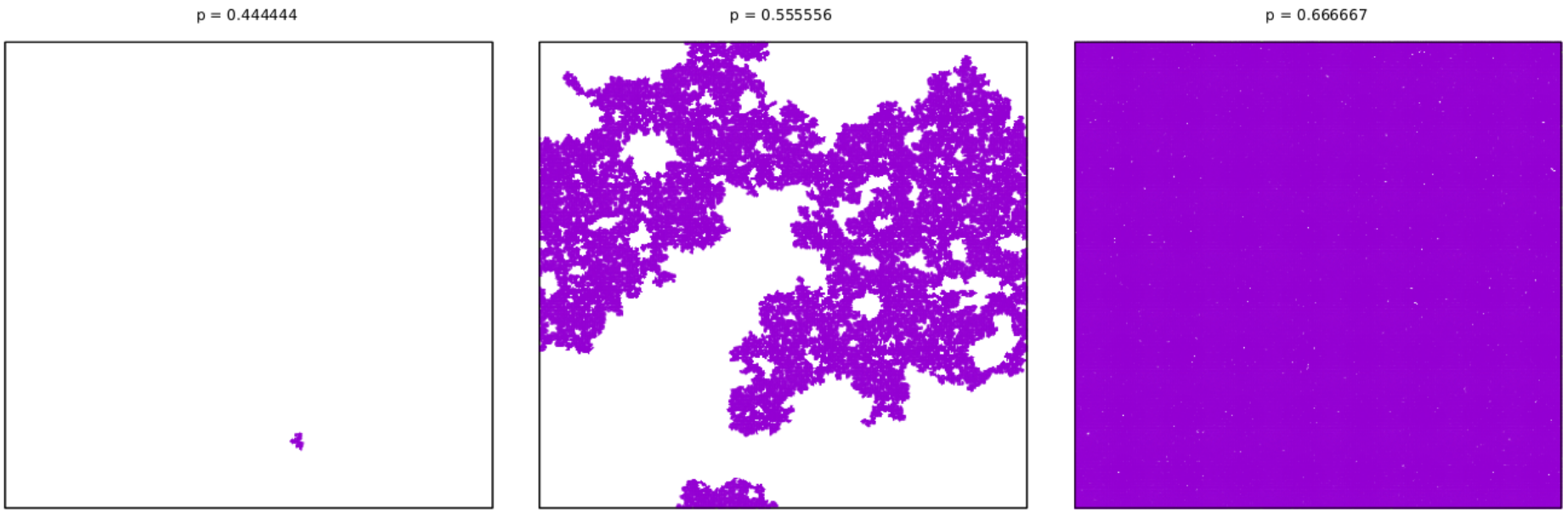
$$x = s \bmod L$$
$$y = s \backslash L,$$

where the function mod returns the remainder of s/L and $\backslash$ is the integer division.

Results

Results

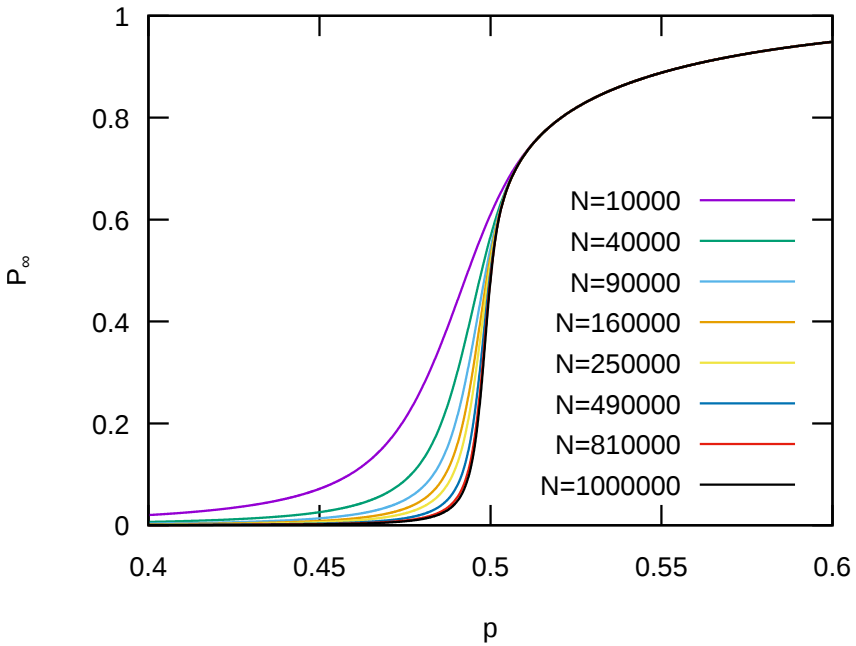
In the following slides you will see the main results that you are expected to obtain. Let's start with lattices.

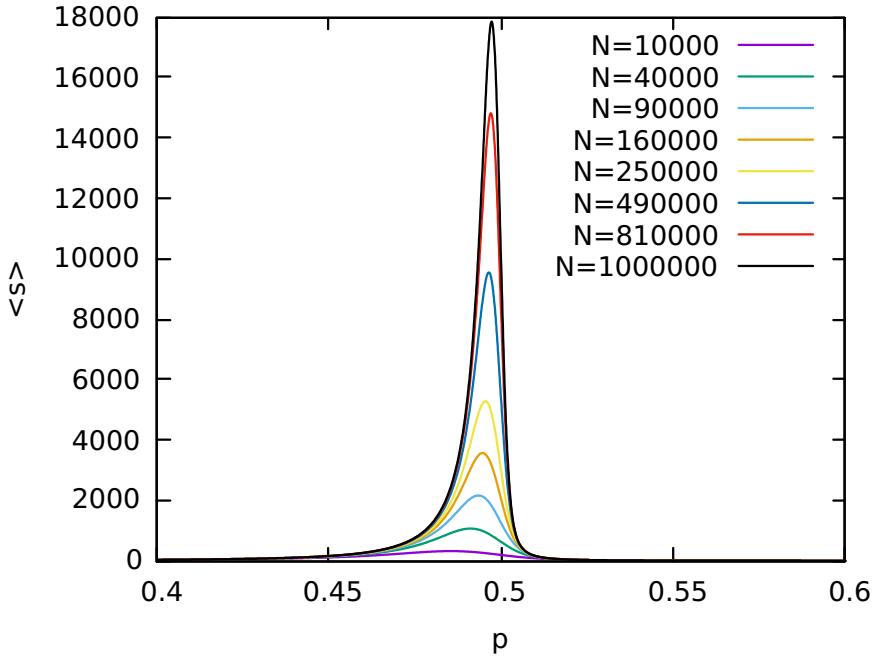


These are the *pictures* taken at different p (this is defined as the number of activated bonds divided by the total number of bonds). This figure shows the growth of the largest cluster in the square lattice.

P_∞ and $\langle s \rangle$

These are P_∞ and $\langle s \rangle$ for the square lattice.





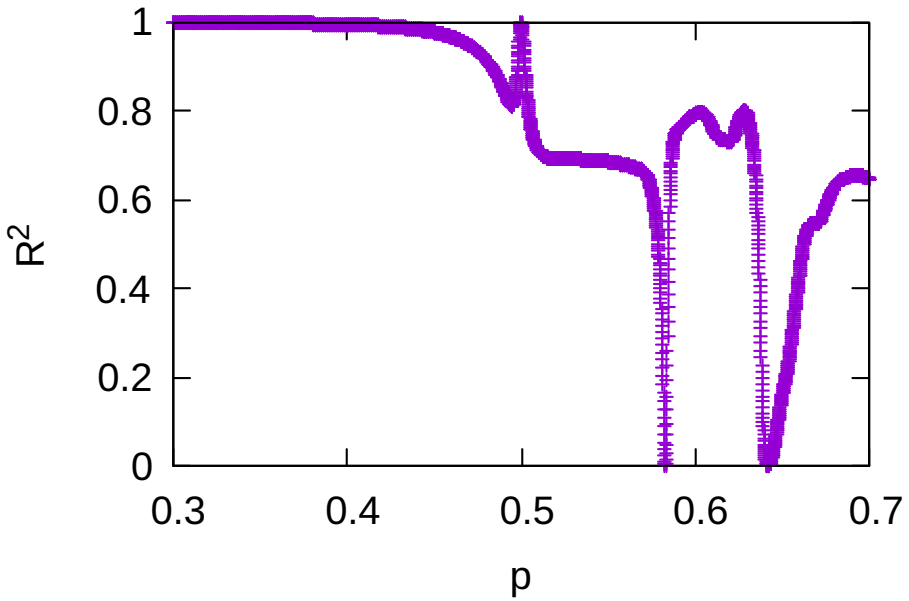
Finite size scaling

To get the probability p_c and the critical exponents you need to use Finite-Size Scaling (FSS):
Around the phase transition, the giant component goes as

$$P_\infty \propto \xi^{-\beta/\nu}.$$

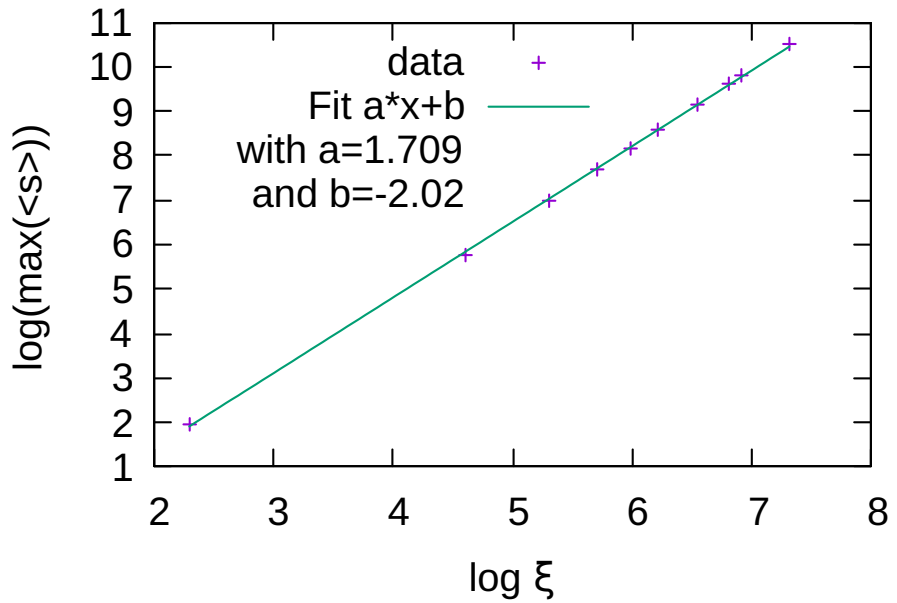
Take the data obtained from different lattice sizes and plot P_∞ as a function of $\xi = \sqrt{N}$ for only one value of p in a log-log scale.

Now fit a straight line to the data points and compute the coefficient of correlation R^2. Repeat for all the values of p. For which p the value of R^2 is larger?



Finite size scaling

The slope of the trend line that gave you an R^2 closer to one is equal to $-\beta/\nu$. Similarly, you can find γ/ν by plotting the maximum of $\langle s \rangle$ as a function of $\xi = \sqrt{N}$ in a log-log scale.



Fit a straight line to the data. The slope of that line is γ/ν.
If I only use the largest lattices I get $\gamma/\nu = 1.75$, which is more accurate.

p_c and the critical exponents for lattices

Using FSS (see the instructions in the Assignment notes) you will find these results:

	Square	Triangular	Honeycomb
p_c	0.5	$2\sin(\pi/18)$	$1-2\sin(\pi/18)$
β	5/36	5/36	5/36
γ	43/18	43/18	43/18
ν	4/3	4/3	4/3

The critical exponents do not depend on the lattice, only on the dimension.

Random networks

For the random networks the percolation threshold goes as

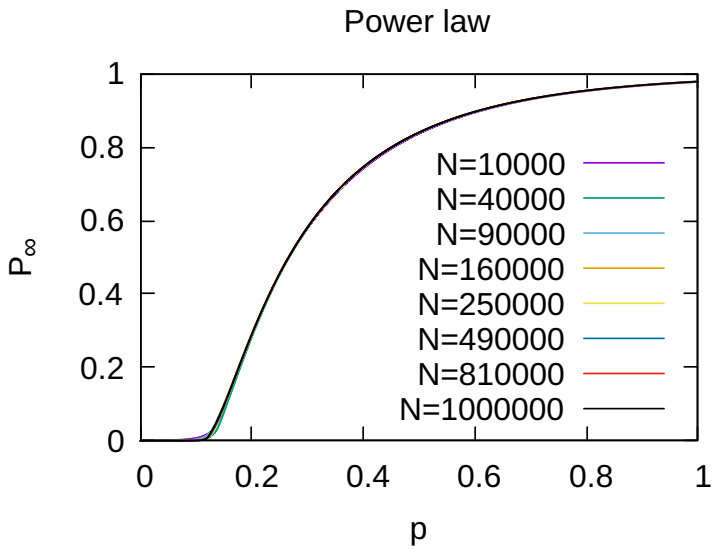
$$p_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle} \tag{1}$$

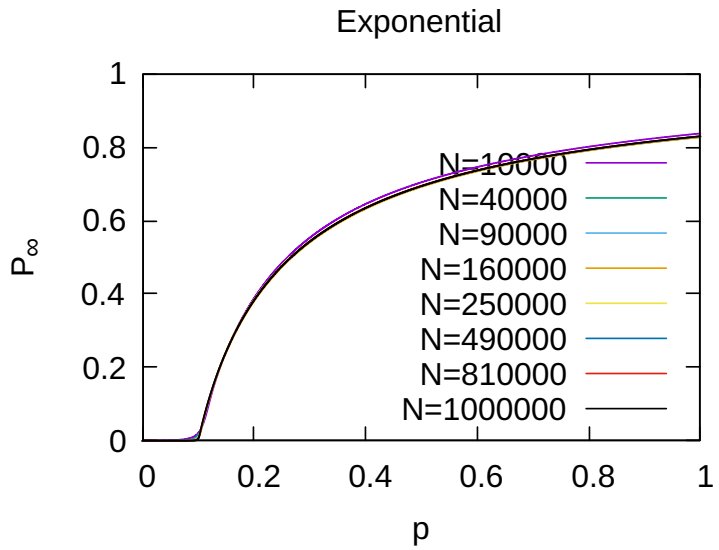
The critical exponents for the power-law random network are also known:

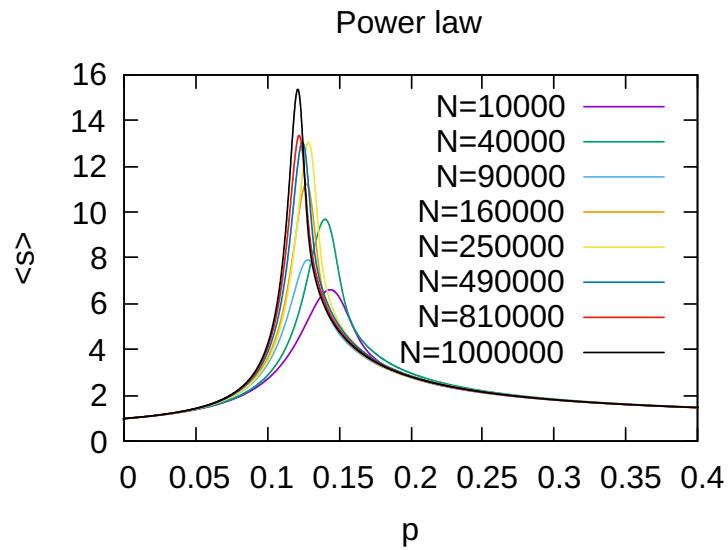
$$\beta = 2, \qquad \gamma = 1, \qquad \bar{\nu} = 5,$$

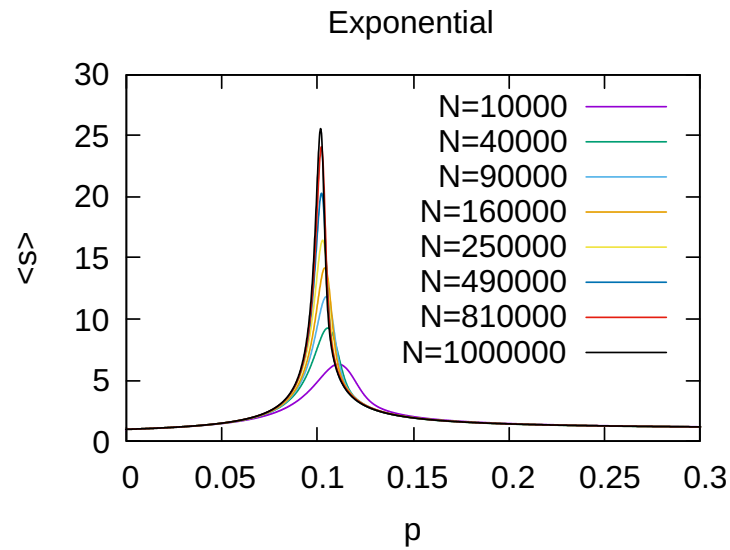
where we have used $\bar{\nu}$ instead of ν. This is defined as $d \cdot \nu$, where d is the dimension. The reason is that for random networks the dimension is not well defined.

And here are some plots









Analytical results

For the random networks it is possible to obtain analytical results. The easiest to find is P_∞.

$$y(p) = 1 - \sum_k \frac{kP(k)}{\langle k \rangle}(1 - py(p))^{k-1}$$

$$P_\infty(p) = 1 - \sum_k P(k)(1 - py(p))^k.$$

Solve the first equation with a root-finding algorithm (bisection, Newton-Raphson, secant method, etc) and put the result into the second equation.

Analytical results