Feuille de TP No 2
Advanced Algorithms
Blerina Sinaimeri

Mars 2017

Exercice 1. You have seen in class the algorithm of Havel-Hakimi for determining whether a given sequence of degrees is graphical. Implement this function in python.

Solution 1.

```
def Havel_Hakimi(sequence):
#check if all elements of the sequence are 0
  if sequence.count(0) == len(sequence):
    return True
#sort the sequence from the biggest value to the smallest one
 sequence.sort(reverse=True)
#if you get a negative degree then return false
 if sequence[len(sequence)-1] < 0: return False
#if the sum of degrees is not even then return false
 if sum(sequence)%2 != 0: return False
#if you get a maximum degree bigger than the number of vertices return false
 if sequence[0] >= len(sequence): return False
#count keeps d1
 count = sequence.pop(0)
#update the sequence
 for i in range(count):
    sequence[i] = sequence[i] - 1
 return Havel_Hakimi(sequence)
```

For the next exercises you should use the package igraph of R. For a quick tutorial see `http://igraph.org/r/doc/`, `http://www.dil.univ-mrs.fr/~tichit/rb/tp1/igraph_tutorial.html`.

Exercice 2.

— Generate and visualize Erdos-Renyi random graphs $G(n, p)$ for $n = \{10, 50, 100, 500\}$ and $p \in \{0.1, 0.5, 0.75\}$.

— Compute the degree distribution of each one of the graphs with the help of the function *degree()*. Plot these distributions in the same graphic. What can you say ?

— For each one of the previously generated graphs calculate the cluster coefficient.

Solution 2.

— To generate random Erdos-Renyi graphs you should use the function *g=erdos.renyi.game()*. See the igraph documentation on how this function works. In our case we should call *g=erdos.renyi.game(10, 0.1 , type=gnp")*. Some examples are given in Fig.1-3.

— Observe that *degree()* gives you the degrees for each vertex. We would like to plot the degree distribution for the graph that is $f_1/n, \ldots, f_{n-1}/n$ where f_i is the number of vertices v such that $deg(v) = i$. This can be done either by computing the occurences of each degree value given by *degree()* or simply by the function *degree_distribution*. We observe that for n large the degree distribution looks very much like a binomial distribution. This displays what we have said in class that for a graph $G(n, p)$ the probabability that a given vertex v has degree equal to d follows a binomial distribution. In Fig. **??** we show the degree distribution for some random graphs together with the binomial distribution for that case (for example for the graph $G(500, 0.75)$, we show its degre distribution and draw the binomial distribution using *dbinom(x,500,0.75)*.

— We can calculate the cluster coefficient in different ways. For example using the function *count_triangles* we get for each vertex the number of triangles it belongs to. Hence, to get the cluster coefficient of a vertex v we have to divide the number of triangles it belongs diveded by $1/2deg(v)[deg(v) - 1]$. For the overall clustering coefficient of a graph we have to sum this value over all vertices.

Now we want to show that the degree distribution of the Erdos-Renyi random graphs is not realistic. To illustrate this we will consider a well-known dataset : the *Zacharys karate club network*. In this dataset vertices are people who were members of a particular university karate club, and two people are connected if they were friends outside the club. This network is described here `https://cran.r-project.org/web/packages/`

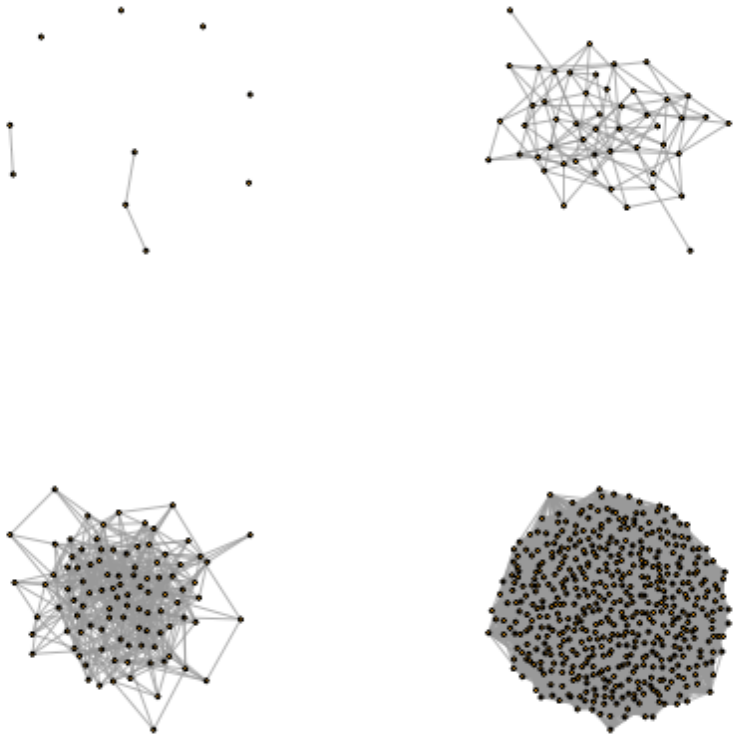


FIGURE 1 – $G(n, p)$ for $p = 0.1$ and $n \in \{10, 50, 100, 500\}$.

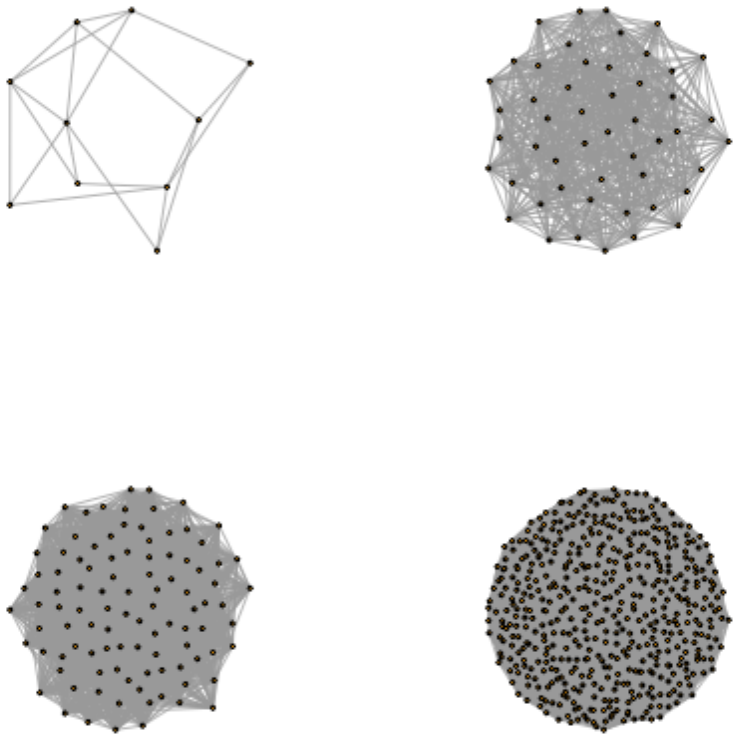


FIGURE 2 – $G(n, p)$ for $p = 0.5$ and $n \in \{10, 50, 100, 500\}$.

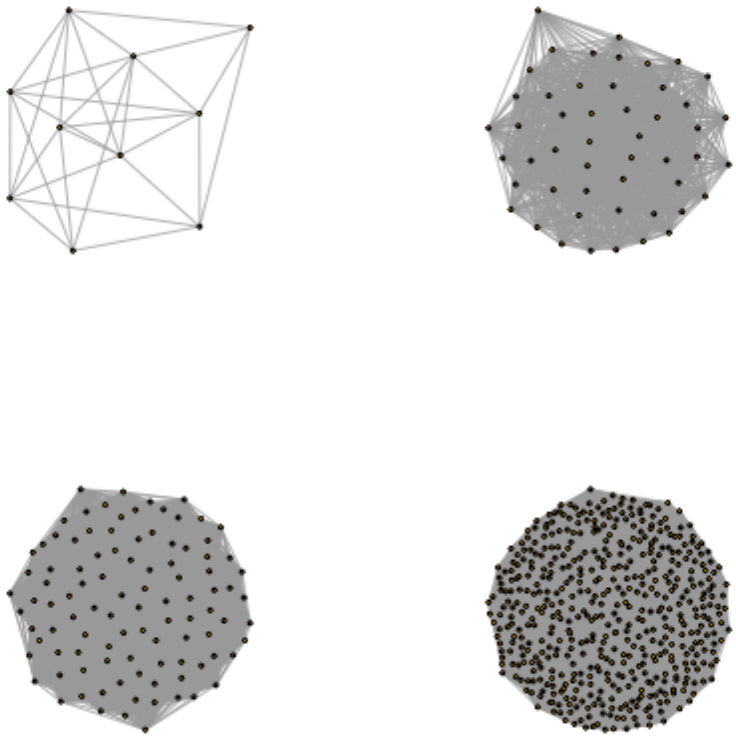
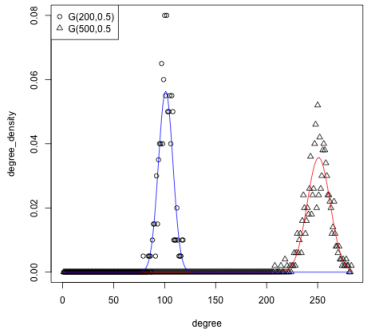
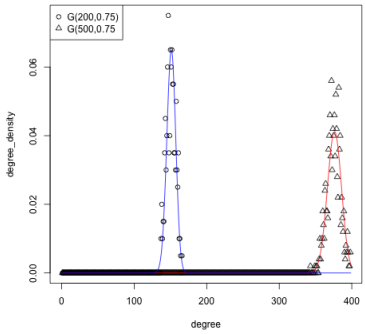


FIGURE 3 – $G(n, p)$ for $p = 0.75$ and $n \in \{10, 50, 100, 500\}$.



(a)

FIGURE 4 – The degree distribution for the graphs $G(200, 0.5), G(500, 0.5)$ together with the binomial distribution.



(a)

FIGURE 5 – The degree distribution for the graphs $G(200, 0.5), G(500, 0.5)$ together with the binomial distribution.

`igraphdata/igraphdata.pdf` and can be downloaded from this website `http://www-personal.umich.edu/~mejn/netdata/`.

Exercice 3.

— Download the dataset "karate" from this website `http://www-personal.umich.edu/~mejn/netdata/` and read it using

$$karate <- read.graph("/MYPATH/karate.gml", format = "gml")$$

Then plot the graph using $plot()$.

— How many edges, vertices has the graph ? Using the function *degree*() get all the degrees of the graph. What is the maximum degree ? And the minimum ? Which vertices have the maximum degree ? The minimum ? Can you say that the graph is connected or not by looking only at the degrees ?

— Denote by k the average degree for this graph. What is the value of k for this graph ? Consider a random graph $G(n, p)$ such that n is equal to the number of vertices of the karate graph, and p is such that the expected average degree is k. Construct $G(n, p)$ using plot and compare it to the karate graph. What can you say ?

— Plot the distribution of the degrees of the karate dataset. In the same graphic plot a Poisson distribution with parameter λ. How would you choose λ such that the model is as much closer to the data as possible ? What can you say about these two graphics ?

Solution 3.

— Note that the function *plot*() has many nice features. For example you can specify the size of the vertex/edges, the colour etc. For example with the following commands I can plot the graph where vertices have the size depend on the value of their degree.

```
> deg <- degree(karate, mode="all")
> plot(karate, vertex.size=deg*3)
```

— You can get the number of vertices and edges using simply the function *summary(karate)*. The graph contructed from the dataset karate has 34 vertices and 78 edges and is not directed. Note that in general it is not possible to say whether the graph is connected by looking only to the degrees. For example for the degree sequence $(2, 2, 2, 2, 2, 2)$ you can have either a cycle on 6 vertices (and in this case the graph is connected) or two triangles (and in this case the graph is not connected). However, in some particular cases you can say something, for example if there is a vertex of degree 0 then the graph is disconnected.

— Recall that the avarage degree of the graph is given by $\frac{1}{n}\sum_v deg(v)$ where $n = |V|$. For this you could either use the function *degree*, or simply you know that $\sum_v deg(v) = 2|E| = 2 \times 78$ and thus $k = 156/32 \approx 4,59$. If we want to model the karate graph using an Erdos-Renyi graph, we should at least generate a graph that has the same

expected average degree. We have seen that the expected average degree for $G(n,p)$ is np thus we should create a graph with $p \approx \frac{k}{n} = 0.13$. In Fig.6 we show both graphs for karate and $G(34, 0.13)$. What can you say looking at these two graphs and their degrees ? For example we can say that in the karate dataset several vertices (like 1,33, 34) have very high degree, while most of the vertices have low degree. This is not observed in the random graph.

— We have to use $\lambda = k$ to get the Poisson model as much closer to the data as possible. In this case it means setting their densities or their mean degrees to be equal. In Fig.7 we plot the degree distribution for the karate dataset together with the Poisson distribution for $\lambda = k$. We observe that the distributions are somehow similar in particular for small values of the degrees. However, at lareg values of degrees the distributions disagree. For example, the Poisson distribution, the probability for a vertex to have degree greater than 16 is around 0.00000675, while in the karate dataset there are two nodes with this degree. Thus, empirically we can say that the degree distributions in reality are often heavy tailed. Thus, the high degree vertices appear much more often than we would expect.

```
> deg <- degree(karate, mode="all")
> deg.dist <- degree_distribution(karate, mode="all")
>  x<- 0:max(deg)
> plot(x, deg.dist, pch=19, cex=1.2,
col="orange", xlab="Degree", ylab="Distribution Frequency")
> lines(x, dpois(x,lambda=4.59),col="red")
```

Exercice 4. Suppose I am interested in the triangles in my graph (i.e. three vertices all connected between them). Use the function *count_triangles* of the package igraph to count the total number of triangles in the graph created from the dataset karate. Then create many random graphs from $G(n,p)$ with p such that the expected average degree is k (like in part (3) of exercise 3) and for each one of them count the number of triangles. Plot a histogram of this distribution. What do you think about the number of triangles in the karate dataset compared to this distribution ?

Solution 4. Hint : For example using the function *count_triangles* we get for each vertex the number of triangles it belongs to. We can also use the function *triangles*.

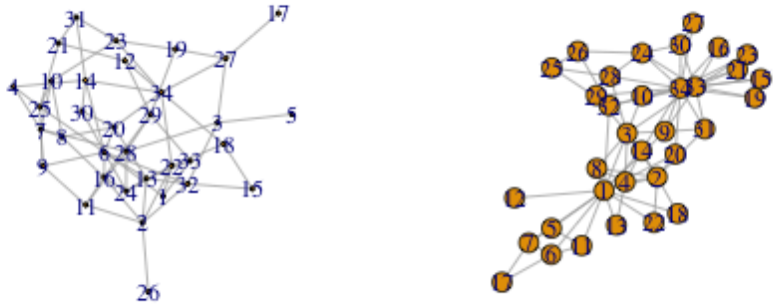


FIGURE 6 – The graph constructed from the karate dataset and a random graph generated from $G(34, 0.13)$.

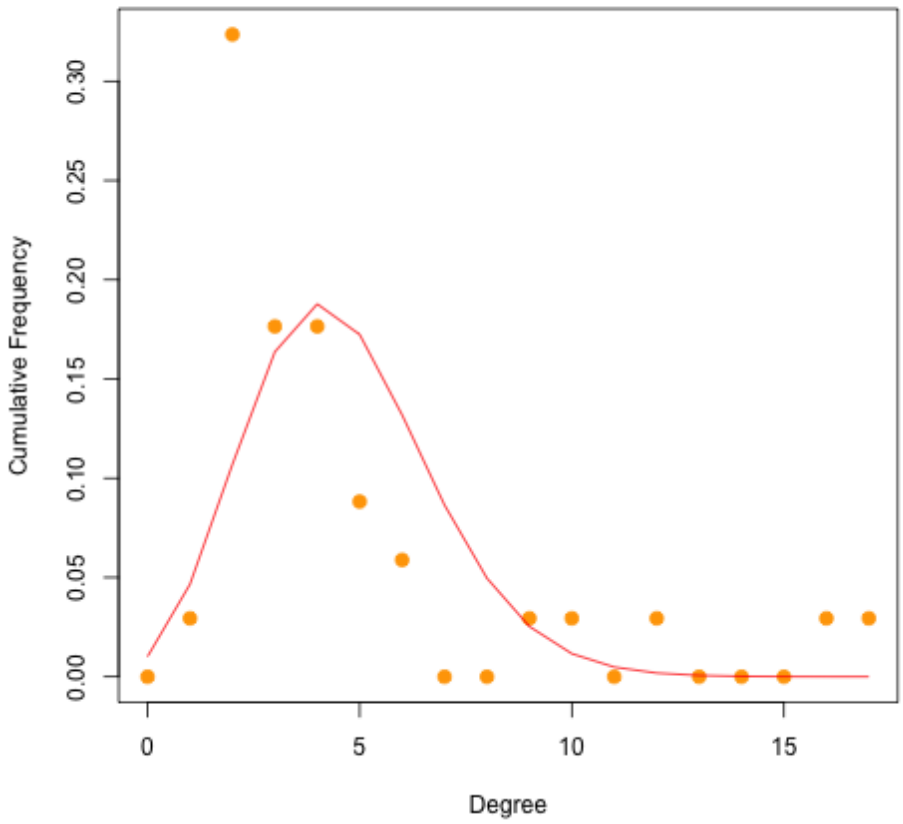


FIGURE 7 – The degree distribution for the karate dataset together with the Poisson distribution for $\lambda = k$.