Graph Models

Contents

Graph Models **1**

Erdos-Reni random graph model 1

Barabasi-Albert model. Preferential attachement 10

Small world model. Watts-Strogats model 15

Model comparison 17

```
library('igraph')
```

Graph Models

Erdos-Reni random graph model

Graph $G\{E, V\}$, nodes $n = |V|$, edges $m = |E|$

$G_{n,p}$ - each pair out of $N = \frac{n(n-1)}{2}$ is connected with probability p, m - random number.

$$\langle m \rangle = p\frac{n(n-1)}{2}$$

$$\langle k \rangle = \frac{1}{n}\sum_i k_i = \frac{2\langle m \rangle}{n} = p(n-1) \approx pn$$

$$\rho = \frac{\langle m \rangle}{n(n-1)/2} = p$$

Probability that i-th node has a degree $k_i = k$ (Bernoulli distribution):

$$P(k_i = k) = P(k) = C_{n-1}^k p^k (1-p)^{n-1-k}$$

- p^k - probability that connects to k nodes (has k-edges)
- $(1-p)^{n-1-k}$ - probability that does not connect to any other node
- C_{n-1}^k - number of ways to select k nodes out of all to connect to

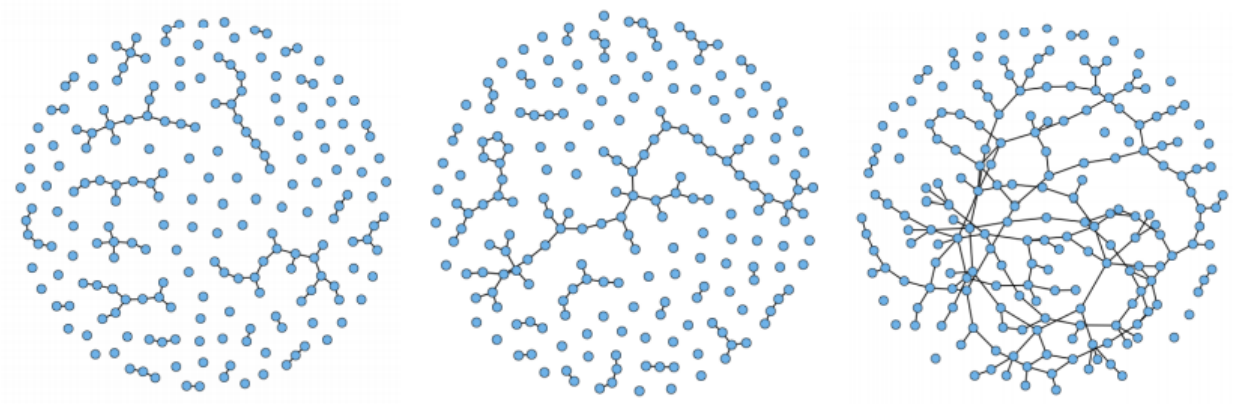
Limiting case of Bernoulli distribution, when $n \to \infty$ at fixed $\langle k \rangle = pn = \lambda$

$$P(k) = \frac{\langle k \rangle^k e^{-\lambda}}{k!}$$

(Poisson distribution)

Phase transition: Consider $G_{n,p}$ as a function of p:

- $p = 0$, empty graph
- $p = 1$, complete (full) graph
- There are exist critical p_c , structural changes from $p < p_c$ to $p > p_c$
- Gigantic connected component appears at $p > p_c$



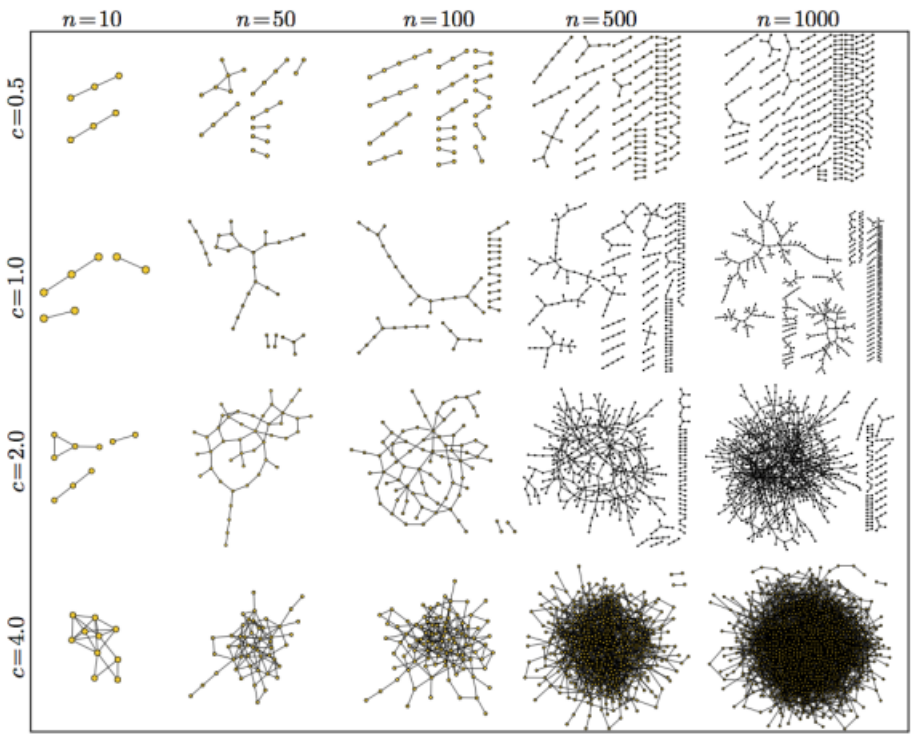
$p < p_c$ $p = p_c$ $p > p_c$

igraph:erdos.renyi.game()

Graph $G(n, p)$, for $n \to \infty$, critical value $p_c = \frac{1}{n}$.

- when $p < p_c(\langle k < 1 \rangle)$ there are no components with more than $O(\ln(n))$ nodes, largest component is a tree
- when $p = p_c(\langle k = 1 \rangle)$ the largest component has $O(n^{2/3})$ nodes
- when $p > p_c(\langle k > 1 \rangle)$ gigantic component has all $O(n)$ nodes

Critical value: $\langle k \rangle = p_c n = 1$ on average one neighbour for a node.



Threshold probabilities

Graph $G(n,p)$.

Threshold probabilities when different subgraphs of k-nodes and l-edges appear in a random graph $p \sim n^{-\frac{k}{l}}$

When

- $p > n^{-k/(k-1)}$, having a tree with k nodes
- $p > n^{-1}$, having a cycle with k nodes
- $p > n^{-2/(k-1)}$, complete subgraph with k nodes

Clustering coefficient

$$C(k) = \frac{numberoflinksbetweenNN}{maxnumberoflinksNN} = \frac{pk(k-1)/2}{k(k-1)/2} = p$$

$$C = p = \frac{\langle k \rangle}{n}$$

When $n \to \infty, C \to 0$.

Graph diameter

$G(n,p)$ is locally tree-like (GCC) (no loops; low clustering coefficient). On average, the number of nodes d steps away from a node $\langle k \rangle^d$. In GCC, around p_c, $\langle k \rangle^d \sim n$,

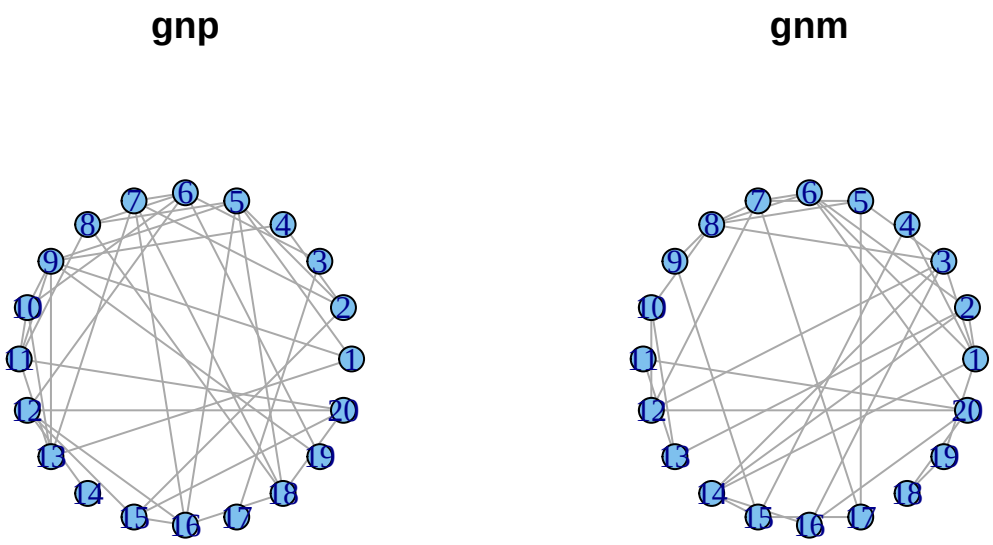
$$d \sim \frac{\ln(n)}{\ln \langle k \rangle}$$

.

Generate random graphs

There is a special function in {igraph} to generate random graphs according to the Erdos-Renyi model. You can specify either the probability for drawing an edge between two arbitrary nodes or the number of edges in the graph:

```
n=20
p=0.2
m=(n*(n-1)/2)*p
er11 <- erdos.renyi.game(n, p, type="gnp")
er12 <- erdos.renyi.game(n, m, type="gnm")
# We will produce two plots and put them in 1 row and 2 columns:
op <- par(mfrow = c(1, 2))
# Produce plots
plot(er11, layout=layout.circle(er11), main="gnp")
plot(er12, layout=layout.circle(er12), main="gnm")
```




```
par(op)
print(length(E((er11))))
```

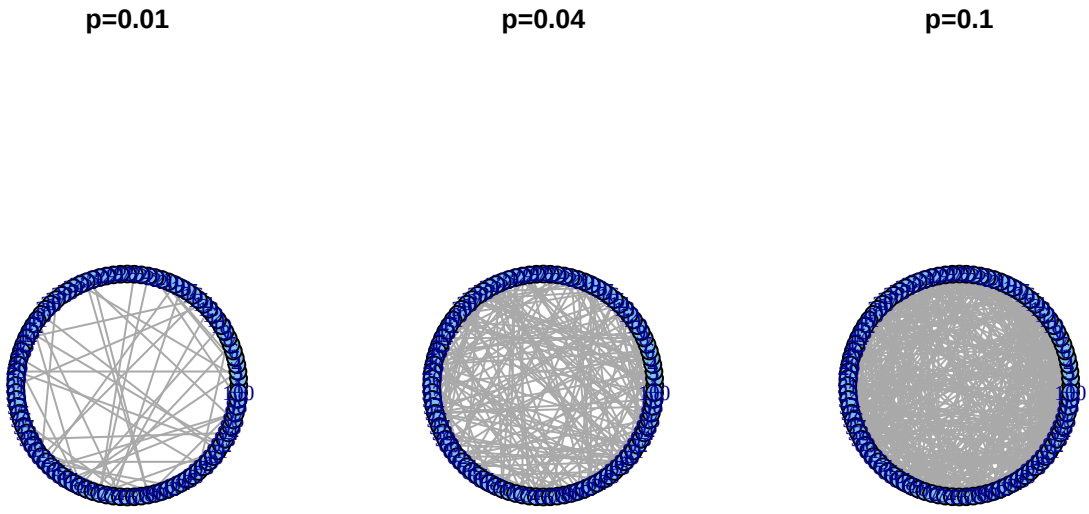
```
## [1] 40
```

```
print(length(E((er12))))
```

```
## [1] 38
```

Now we will slightly change the probability parameter and see how our graph changes:

```
n=100
p1=0.01
p2=0.04
p3=0.1
er21 <- erdos.renyi.game(n, p1, type="gnp")
er22 <- erdos.renyi.game(n, p2, type="gnp")
er23 <- erdos.renyi.game(n, p3, type="gnp")
op <- par(mfrow = c(1, 3))
plot(er21, layout=layout.circle(er21), main="p=0.01")
plot(er22, layout=layout.circle(er22), main="p=0.04")
plot(er23, layout=layout.circle(er23), main="p=0.1")
```




```
par(op)
```

Degree distribution

Now it's your turn - can you compute node degrees for these graphs and print summary statistics?

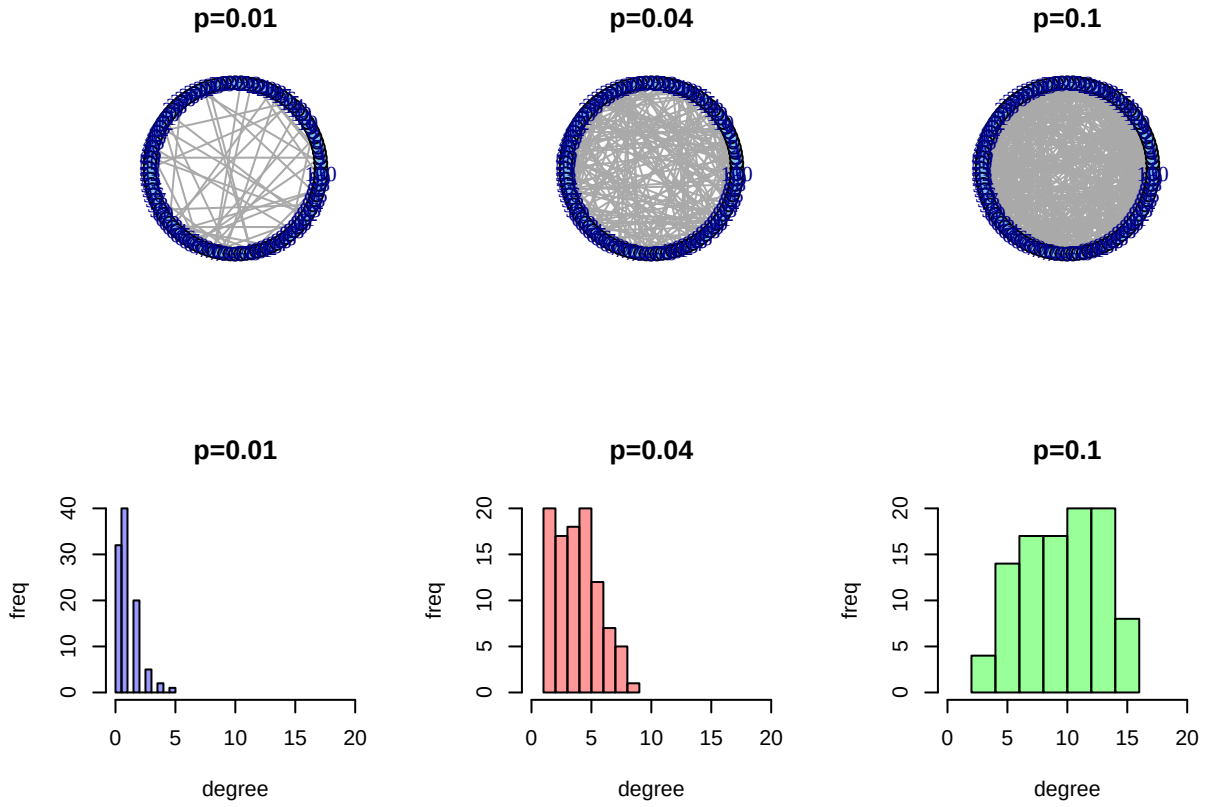
Hints: `your_degrees = degree(your_graph)`

```
der1 <- degree(er21)
summary(der1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    1.00    1.08    2.00    5.00
```

Now we will add degree distributions to the previous plots:

```
der21 <- degree(er21)
der22 <- degree(er22)
der23 <- degree(er23)
op <- par(mfrow = c(2, 3))
plot(er21, layout=layout.circle(er21), main="p=0.01")
plot(er22, layout=layout.circle(er22), main="p=0.04")
plot(er23, layout=layout.circle(er23), main="p=0.1")
hist(der21, col=rgb(0,0,1,.4), xlim=c(0,20), xlab="degree",ylab="freq", main="p=0.01")
hist(der22, col=rgb(1,0,0,.4), xlim=c(0,20), xlab="degree",ylab="freq", main="p=0.04")
hist(der23, col=rgb(0,1,0,.4), xlim=c(0,20), xlab="degree",ylab="freq", main="p=0.1")
```



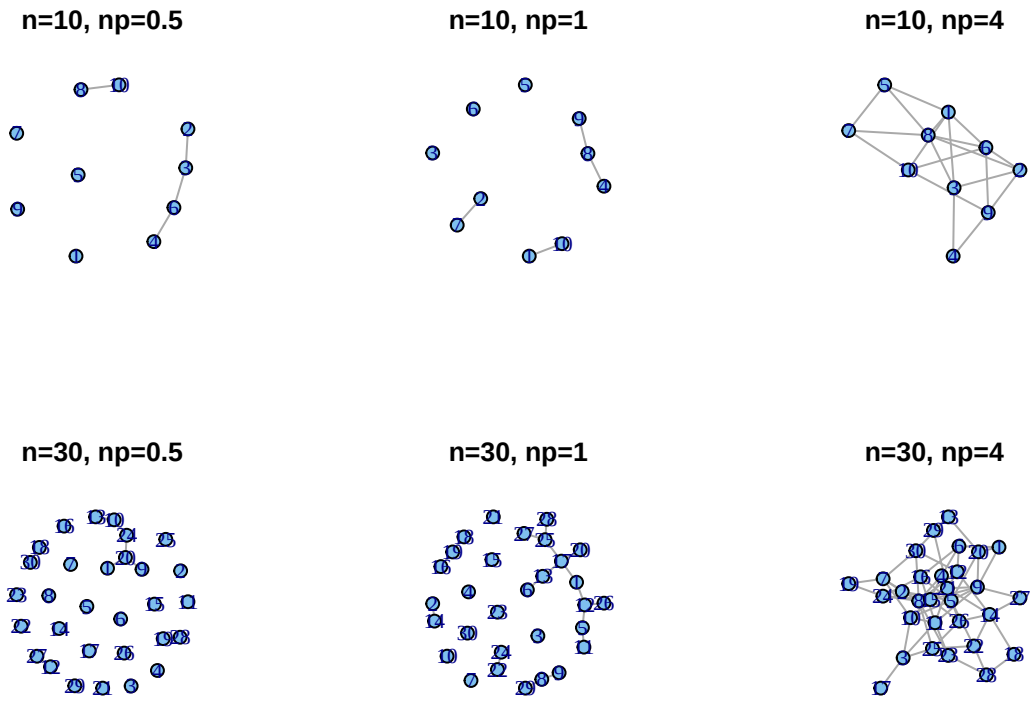

```
par(op)
```

Phase transition and gigantic connected component

Now, let's explore gigantic connected components in random graphs with different probabilities:

```
n1=10
n2=30
#n3=200
er311 <- erdos.renyi.game(n1, 0.5/n1, type="gnp")
er312 <- erdos.renyi.game(n1, 1.1/n1, type="gnp")
er313 <- erdos.renyi.game(n1, 4/n1, type="gnp")
er321 <- erdos.renyi.game(n2, 0.5/n2, type="gnp")
er322 <- erdos.renyi.game(n2, 1.1/n2, type="gnp")
er323 <- erdos.renyi.game(n2, 4/n2, type="gnp")

op <- par(mfrow = c(2, 3))
plot(er311, layout=layout.fruchterman.reingold(er311), main="n=10, np=0.5")
plot(er312, layout=layout.fruchterman.reingold(er312), main="n=10, np=1")
plot(er313, layout=layout.fruchterman.reingold(er313), main="n=10, np=4")
plot(er321, layout=layout.fruchterman.reingold(er321), main="n=30, np=0.5")
plot(er322, layout=layout.fruchterman.reingold(er322), main="n=30, np=1")
plot(er323, layout=layout.fruchterman.reingold(er323), main="n=30, np=4")
```



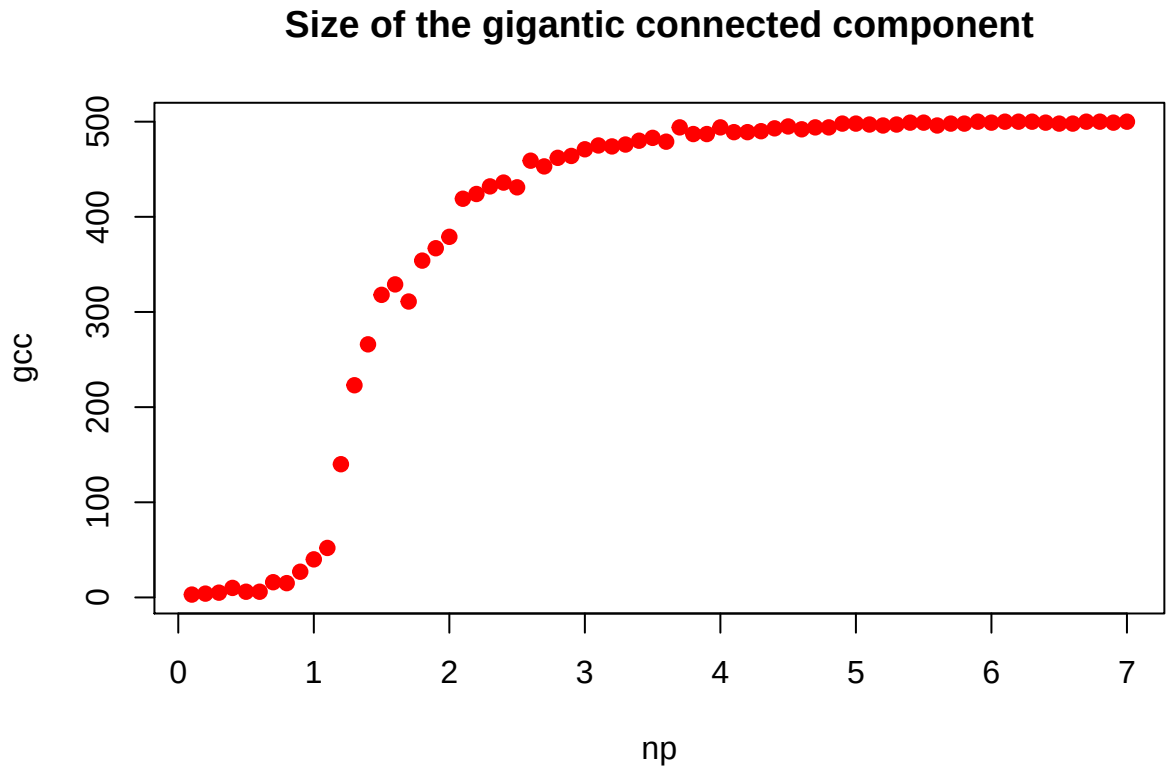

```
par(op)
```

How does the size of the gigantic connected component depend on the probability?

```
gcc=c()
cc=c()
probability=c()
```

```r
for (i in 1:70){
  prob=0.0002*i
  probability=c(probability, prob)
  er <- erdos.renyi.game(500, prob, type="gnp")
  comp <- clusters(er)
  largest <- max(comp$csize)
  gcc=c(gcc, largest)
  clust <- transitivity(er, type="global")
  cc=c(cc, clust)
}

par(mfrow = c(1,1))
plot(probability*500, gcc, pch=19, col=26, xlab="np", main="Size of the gigantic connected component")
```




```r
log(500)
```

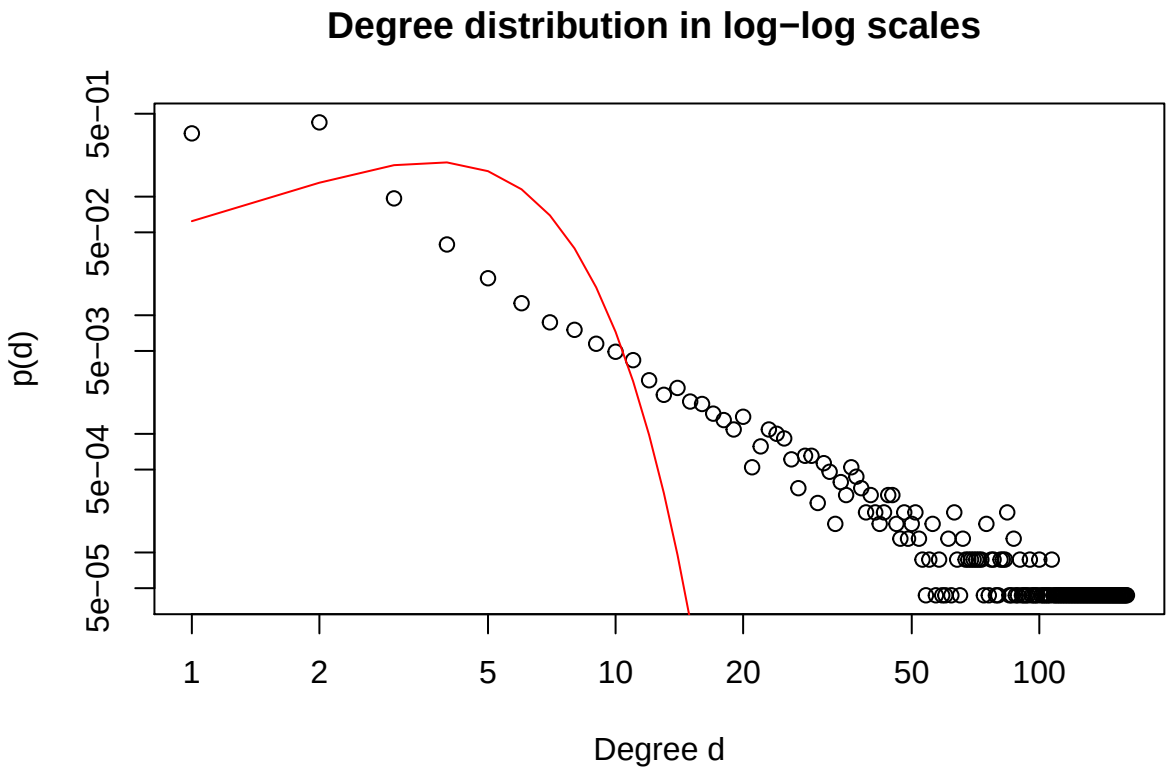
```
## [1] 6.214608
```

Trying to put ER model on real-networks

Load Internet network in R. Plot its degree distribution in log-log scales.

Calculate average degree of the network and put a line for Poisson distribution `dpois()` on the same plot `lines()`

```
g <- read.graph(file = 'as-22july06.gml', format = 'gml')
d <- degree.distribution(g)
plot(d[which (d>0)],log = 'xy', xlab = 'Degree d', ylab = 'p(d)', main = 'Degree distribution in log-log
par(new=TRUE)
avd <- mean(degree(g))
p <- dpois(1:length(d), avd)
lines(p, col="red")
```



Not even close!

To finish with it identife the regime of this network as if it was Random Network. Determine the size of GCC and convice yourself that it conflicts with ER theory.

```
#Compute the size of the GCC for our network:
length(clusters(g)$csize)
```

```
## [1] 1
```

```
# We only have one connected component in this network!
# For a random network of the same size to be fully connected, it should have an average degree of log(
log(vcount(g))
```

```
## [1] 10.04164
```

```
#And our actual average node degree is:
avd
```

```
## [1] 4.218613
```

```
# If our network was a random network with such a low average degree, it wouldn't be fully connected!
#This is also an evidence that our network is not random.
```

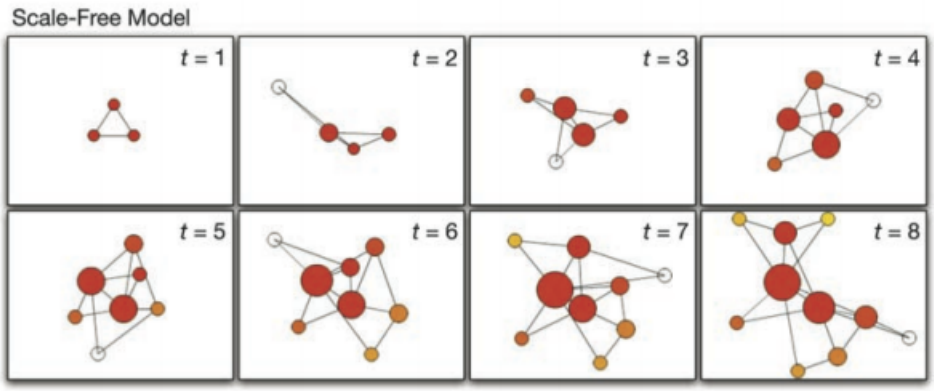
Barabasi-Albert model. Preferential attachement

Dynamical growth model

- $t = 0$, n_0 nodes
- growth: on every step add a node with m_0 edges $m_0 \leq n_0$, $k_i(i) = m_0$
- Preferential attachment: probability of linking to existing node is proportional to the node degree k_i

$$\Pi(k_i) = \frac{k_i}{\sum_i k_i} = \frac{k_i}{2m_0 t}$$

after t steps: $n_0 + t$ nodes, $m_0 t$ edges.



Continues time, mean field approximation:

$$k_i(t + \delta t) = k_i(t) + m_0 \Pi(k_I)\delta t$$

$$\frac{\mathrm{d}k_i(t)}{\mathrm{d}t} = m_0 \Pi(k_i) = \frac{m_0 k_i}{2m_0 t}$$

$$\frac{\mathrm{d}k_i(t)}{\mathrm{d}t} = \frac{k_i(t)}{2t}$$

initial conditions: $k_i(i) = m_0$

$$\int_{m_0}^{k_i(t)} \frac{dk_i}{k_i} = \int_i^t \frac{dt}{2t}$$

Solution:

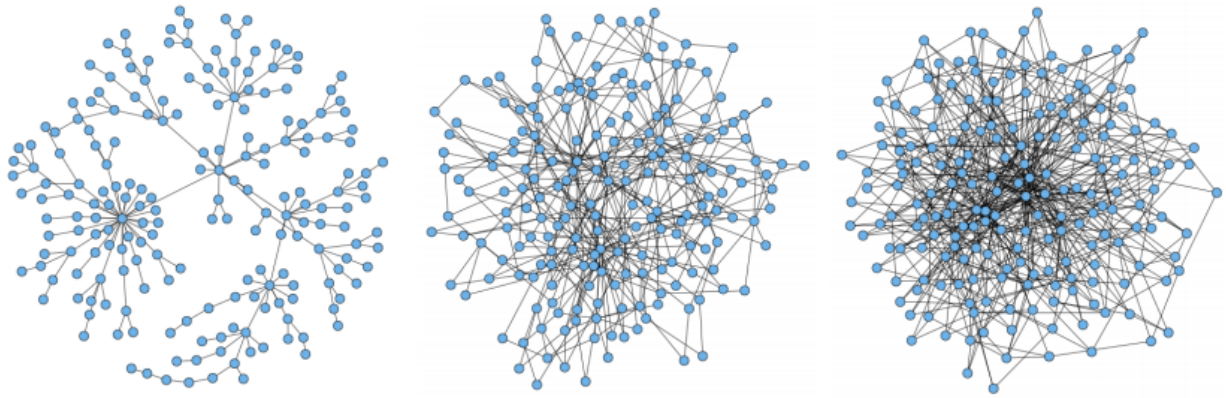
$$k_i(t) = m_0 \left(\frac{t}{i}\right)^{1/2}$$

Time evolution of a node degree:

$$k_i(t) = m_0 \left(\frac{t}{i}\right)^{1/2}$$

Degree distribution function:

$$P(k) = \frac{2m_0^2}{k^3}$$



$m_0 = 1$ $m_0 = 2$ $m_0 = 3$

igraph: barabasi.game()

Power law distribution function:

$$P(k) = \frac{2m_0^2}{k^3}$$

Average path length (analytical result):

$$\langle L \rangle \sim \frac{\log(N)}{\log\log(N)}$$

Clustering coefficient (numerical result):

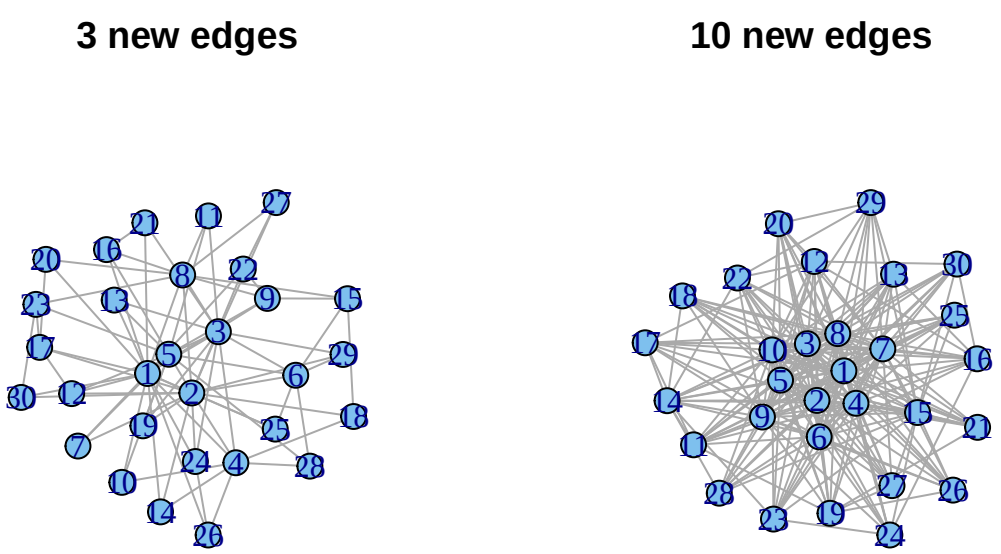
$$C \sim N^{-0.75}$$

Generate Barabasi-Albert graphs

We may use a function from igraph

```
b1=barabasi.game(30, power = 1, m=3, directed=F)
b2=barabasi.game(30, power = 1, m=10, directed=F)

op <- par(mfrow = c(1, 2))
plot(b1, layout=layout.fruchterman.reingold(b1), main="3 new edges")
plot(b2, layout=layout.fruchterman.reingold(b2), main="10 new edges")
```



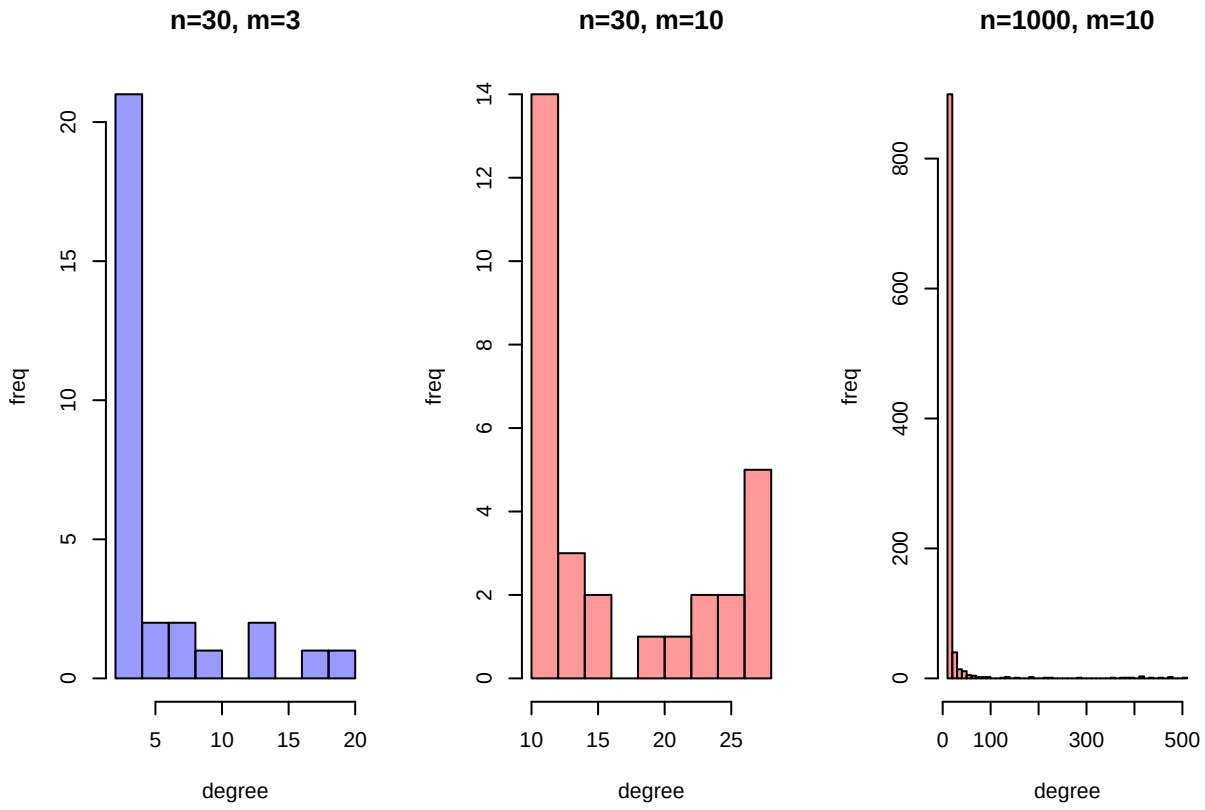

```
par(op)
```

Degree distribution

We conider degree distributions for the Barabasi-Albert graphs. Let's plot degree distributions for the small graphs and for a larger graph and compare the results:

```
b3=barabasi.game(1000, power = 1, m=10, directed=F)

db1 <- degree(b1)
db2 <- degree(b2)
db3 <- degree(b3)
op <- par(mfrow = c(1, 3))
hist(db1, breaks=10, col=rgb(0,0,1,.4), xlab="degree",ylab="freq", main="n=30, m=3")
hist(db2, breaks=10, col=rgb(1,0,0,.4),xlab="degree",ylab="freq", main="n=30, m=10")
hist(db3, breaks=50, col=rgb(1,0,0,.4),xlab="degree",ylab="freq", main="n=1000, m=10")
```




```
par(op)
```

Can you estimate α for the third graph?

Hints:

`power.law.fit`

`fit$alpha`

```
fit <- power.law.fit(degree(b3))
fit$alpha
```

```
## [1] 2.035374
```

Dynamical change

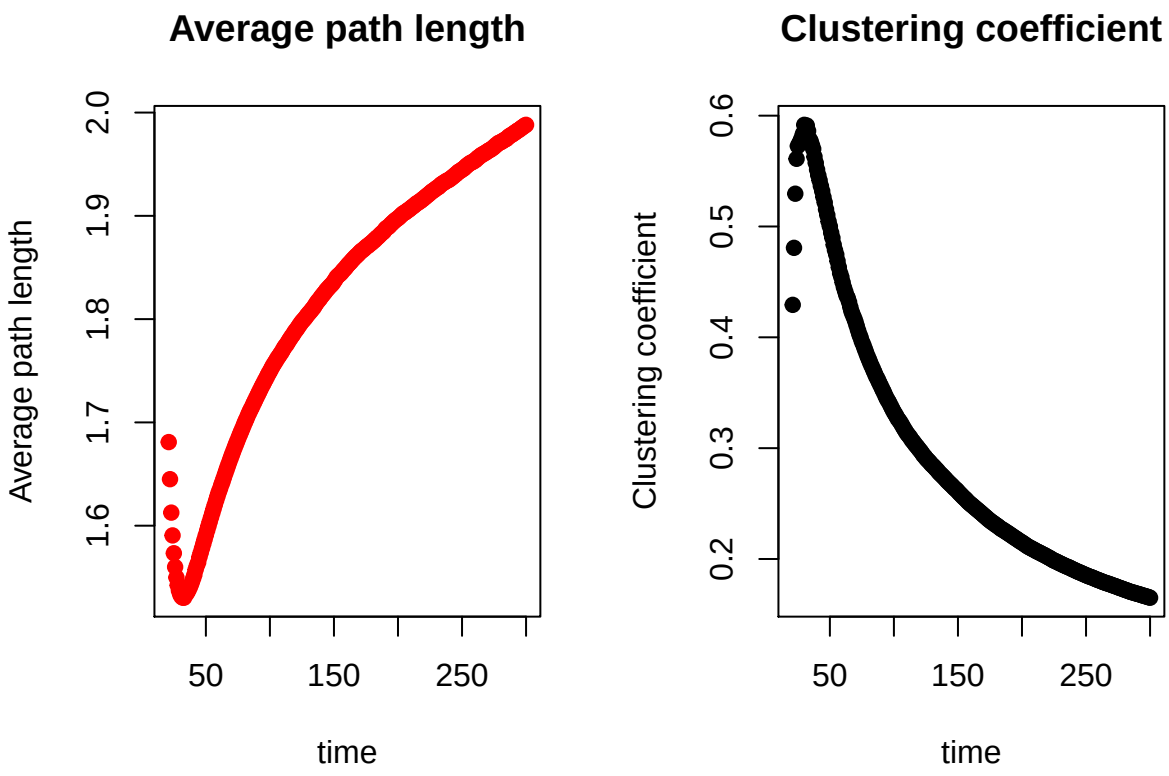
Now, we will look at how clustering coefficient, average path length and node degrees change over time.

```
avpathlen=c()
clustcoef=c()
deg30=c()
deg70=c()
ba=barabasi.game(20, power = 1, m=3, directed=F)
m=15
```

```
for (i in 21:300){
  ba=add.vertices(ba, 1)
  #To select neighbors of the new node, we sample from a vector of existing nodes with a probability se
  neib=sample(seq(1,vcount(ba)), m, prob = degree(ba,normalized = T))
  new_edges=c()
  for (n in neib){
    new_edges=c(new_edges, c(i, n))
  }
  ba=add.edges(ba, new_edges)
  avpathlen=c(avpathlen, average.path.length(ba))
  clustcoef=c(clustcoef, transitivity(ba, type="global"))
  if (i>30){
    deg30=c(deg30, degree(ba, 30))}
  if (i>70){
    deg70=c(deg70, degree(ba, 70))}
  }
```

Let's plot average path length and clustering coefficient against the number of nodes in a graph:

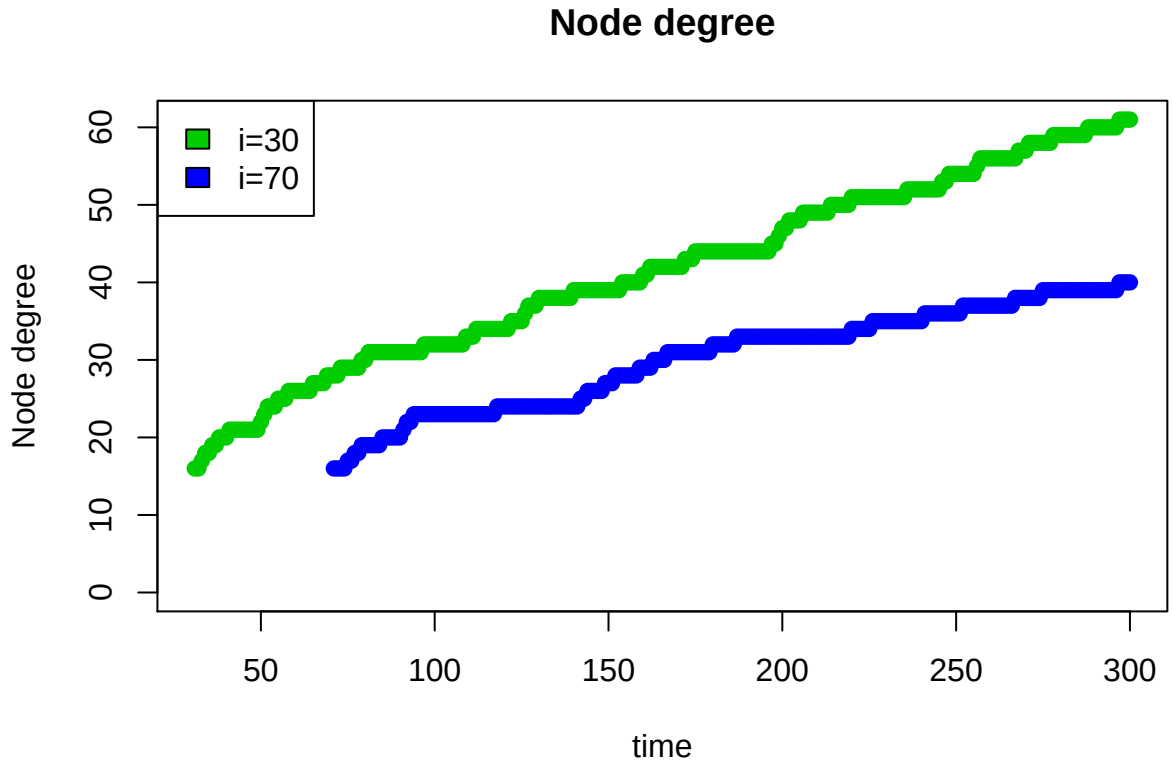
```
xaxis=c(21:300)
op <- par(mfrow = c(1, 2))
plot(xaxis, avpathlen, pch=19, col=26, xlab="time", ylab="Average path length", main="Average path leng
plot(xaxis, clustcoef, pch=19, col=49, xlab="time", ylab="Clustering coefficient", main="Clustering coe
```



```r
par(op)
```

And now we may see the node degrees:

```r
plot(c(31:300), deg30, pch=19, col=35, ylim=c(0, max(deg30)), xlab="time", ylab="Node degree", main="No
points(c(71:300), deg70, pch=19, col=84)
#We put the legend to the top left corner and fill the markers with the same colors as on the main grap
legend("topleft", c("i=30", "i=70"), fill=c(35, 84))
```



Small world model. Watts-Strogats model

Single parameter model, interpolation between regular lattice and random graph

- start with regular lattice with n nodes, k edges per vertex (node degree), $k << n$
- randomly connect with other nodes with probability p, forms $pnk/2$ "long distance" connections from total of $nk/2$ edges
- $p = 0$ regular lattice, $p = 1$ random graph

Node degree distribution: Poisson like

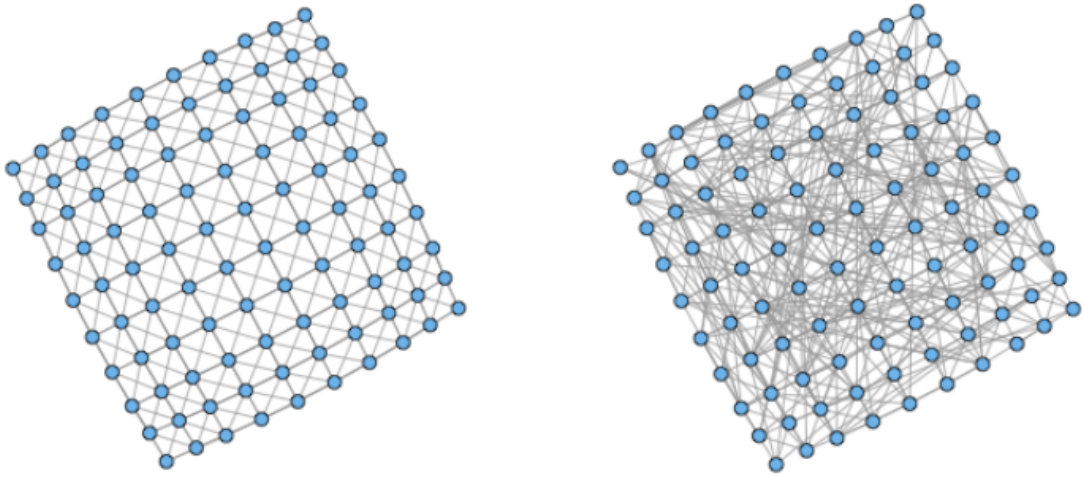
Ave. path length $\langle L(p) \rangle$

- $p \to 0$, ring lattice, $\langle L(0) \rangle = \frac{2n}{k}$

- $p \to 1$ random graph, $\langle L(1) \rangle = \frac{\log(n)}{\log(k)}$

Clustering coefficient C(p):

- $p \to 0$, ring lattice, $C(0) = 3/4 = const$
- $p \to 1$, random graph, $C(1) = \frac{k}{n}$



20% rewiring:
ave. path length = 3.58 → ave. path length = 2.32
clust. coeff = 0.49 → clust. coeff = 0.19
igraph:watts.strogatz.game()

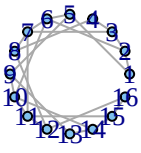
Generate small world graphs

We will use a function from igraph package:

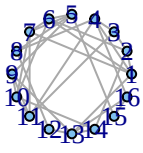
```
lat <- graph.lattice(length=4, dim=2, nei=1)
ws1 <- watts.strogatz.game(dim=2, size=4, nei=1, p=0.1, loops = FALSE, multiple = FALSE)
ws2 <- watts.strogatz.game(dim=2, size=4, nei=1, p=0.4, loops = FALSE, multiple = FALSE)
ws3 <- watts.strogatz.game(dim=2, size=4, nei=1, p=0.9, loops = FALSE, multiple = FALSE)

op <- par(mfrow = c(2, 2))
plot(lat, layout=layout.circle(lat), main="Lattice")
plot(ws1, layout=layout.circle(ws1), main="Rewiring probability 0.1")
plot(ws2, layout=layout.circle(ws2), main="Rewiring probability 0.4")
plot(ws3, layout=layout.circle(ws3), main="Rewiring probability 0.9")
```

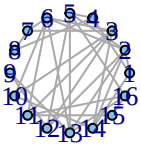
Lattice



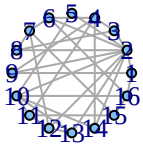
Rewiring probability 0.1



Rewiring probability 0.4



Rewiring probability 0.9



```
par(op)
```

Now it's your turn again. Can you compute clustering coefficients and average path lengths for our small world graphs?

Hints:

`transitivity(your_graph, type="global")`

`average.path.length(your_graph)`

Can you plot a degree distribution for these graphs?

Hints:

See above - we've already plotted degree distribution for the Internet network today.

Model comparison

	Random	BA model	WS model	Empirical networks
P(k)	$\frac{\lambda^k e^{-\lambda}}{k!}$	k^{-3}	poisson like	power law
C	$\langle k \rangle / N$	$N^{-0.75}$	const	large
$\langle L \rangle$	$\frac{\log(N)}{\log(\langle k \rangle)}$	$\frac{\log(N)}{\log\log(N)}$	$\log(N)$	small