

Graph Models

Contents

Graph Models	1
Erdos-Reni random graph model	1
Barabasi-Albert model. Preferential attachment	8
Small world model. Watts-Strogats model	12

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

Graph Models

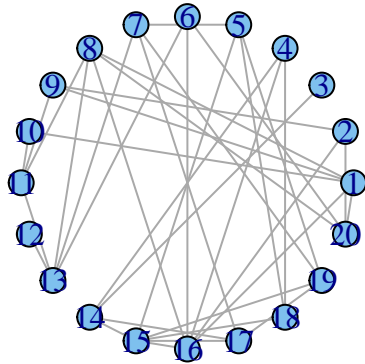
Erdos-Reni random graph model

Generate random graphs

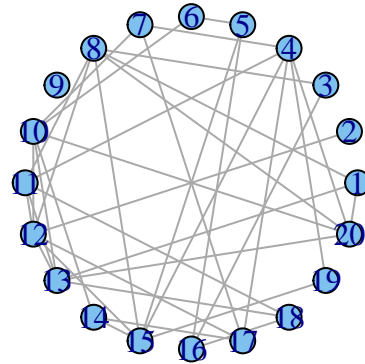
There is a special function in {igraph} to generate random graphs according to the Erdos-Renyi model. You can specify either the probability for drawing an edge between two arbitrary nodes or the number of edges in the graph:

```
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")
```

gnp



gnm



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

```
## [1] 37
```

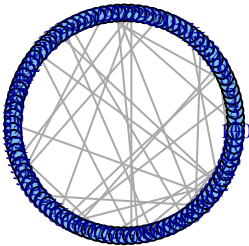
```
print(length(E((er12))))
```

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

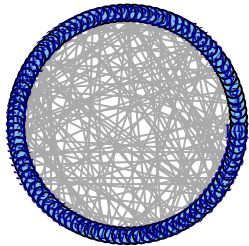
Now we will slightly change the probability parameter and see how our graph changes:

```
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")
```

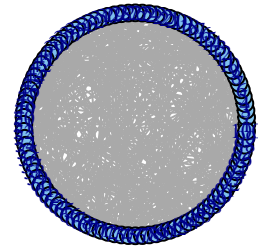
p=0.01



p=0.04



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)`

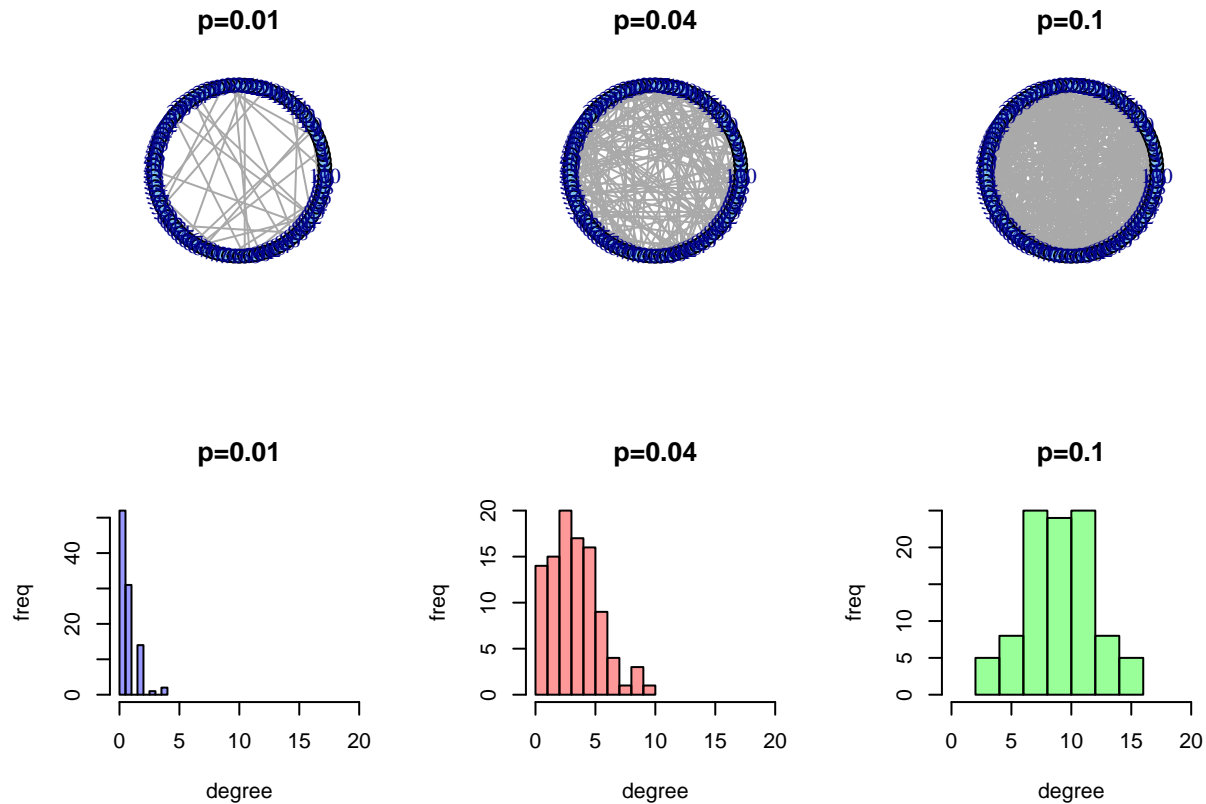
```
der1 <- degree(er21)
summary(der1)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      0.0     0.0     0.0     0.7     1.0     4.0
```

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

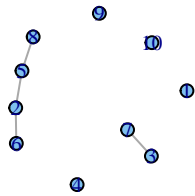
Now, let's explore gigantic connected components in random graphs with different probabilities:

```
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")

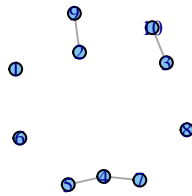
op <- par(mfrow = c(2, 3))
plot(er311, layout=layout.fruchterman.reingold(er311), main="n=10, np=0.5")
plot(er312, layout=layout.fruchterman.reingold(er312), main="n=10, np=1")
plot(er313, layout=layout.fruchterman.reingold(er313), main="n=10, np=4")
plot(er321, layout=layout.fruchterman.reingold(er321),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

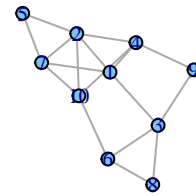
n=10, np=0.5



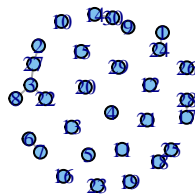
n=10, np=1



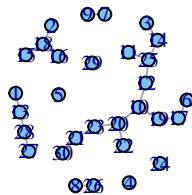
n=10, np=4



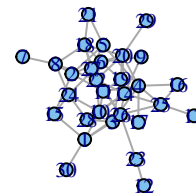
n=30, np=0.5



n=30, np=1



n=30, np=4



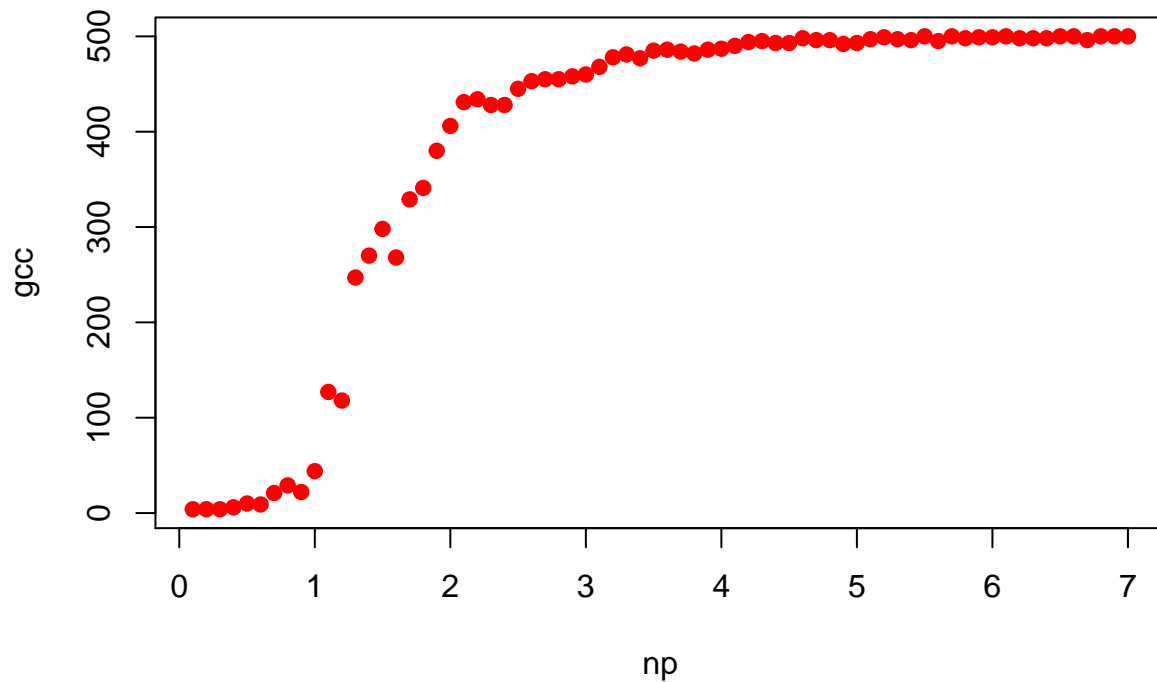
```
par(op)
```

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

```
gcc=c()
cc=c()
probability=c()
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$size)
  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")
```

Size of the gigantic connected component



```
log(500)
```

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

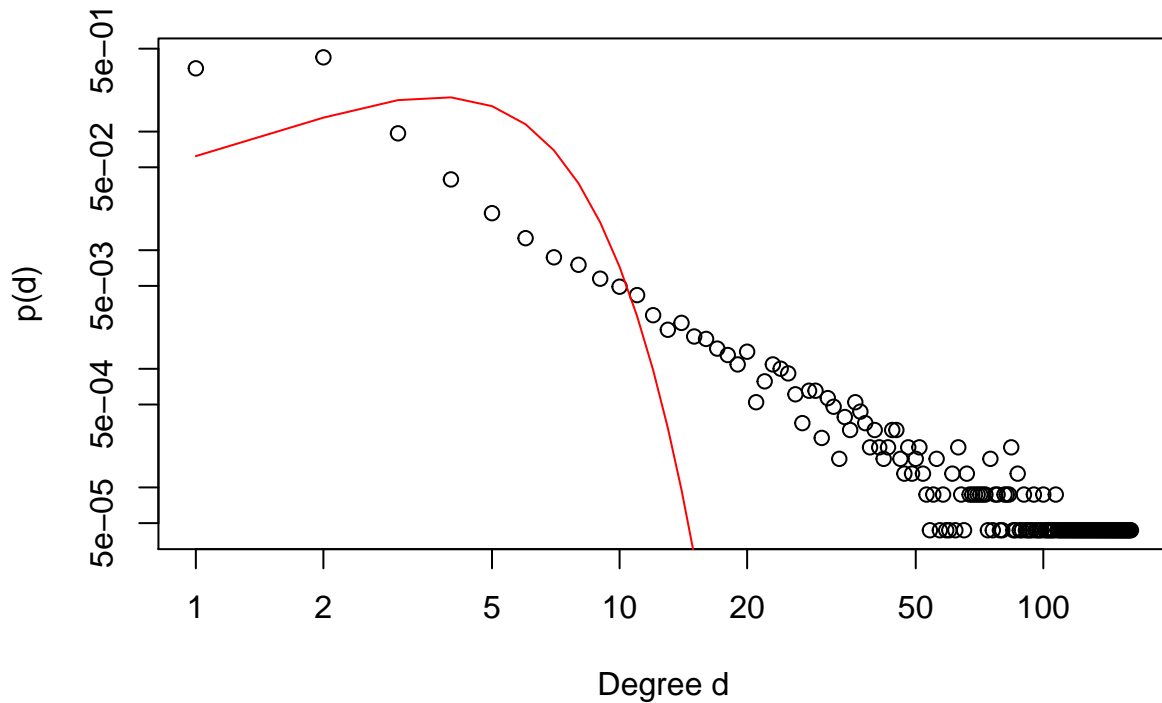
Trying to put ER model on real-networks

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")
```

Degree distribution in log-log scales



Not even close!

To finish with it identify the regime of this network as if it was Random Network. Determine the size of GCC and convince 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(vcount(g))  
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 attachment

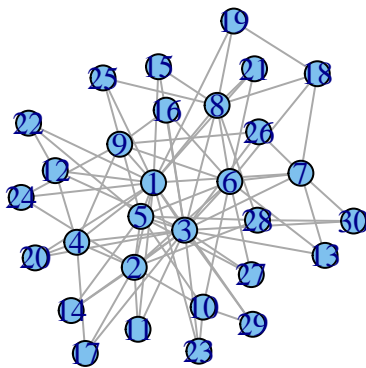
Generate Barabasi-Albert graphs

We may use a function from igraph

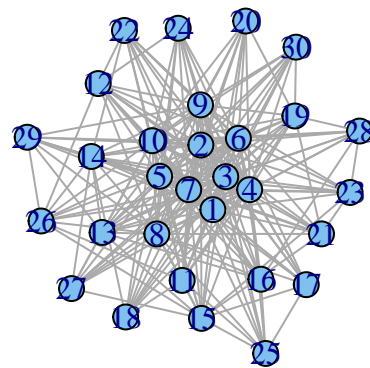
```
b1=barabasi.game(30, power = 1, m=3, directed=F)
b2=barabasi.game(30, power = 1, m=10, directed=F)

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")
```

3 new edges



10 new edges



```
par(op)
```

Degree distribution

We consider 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)

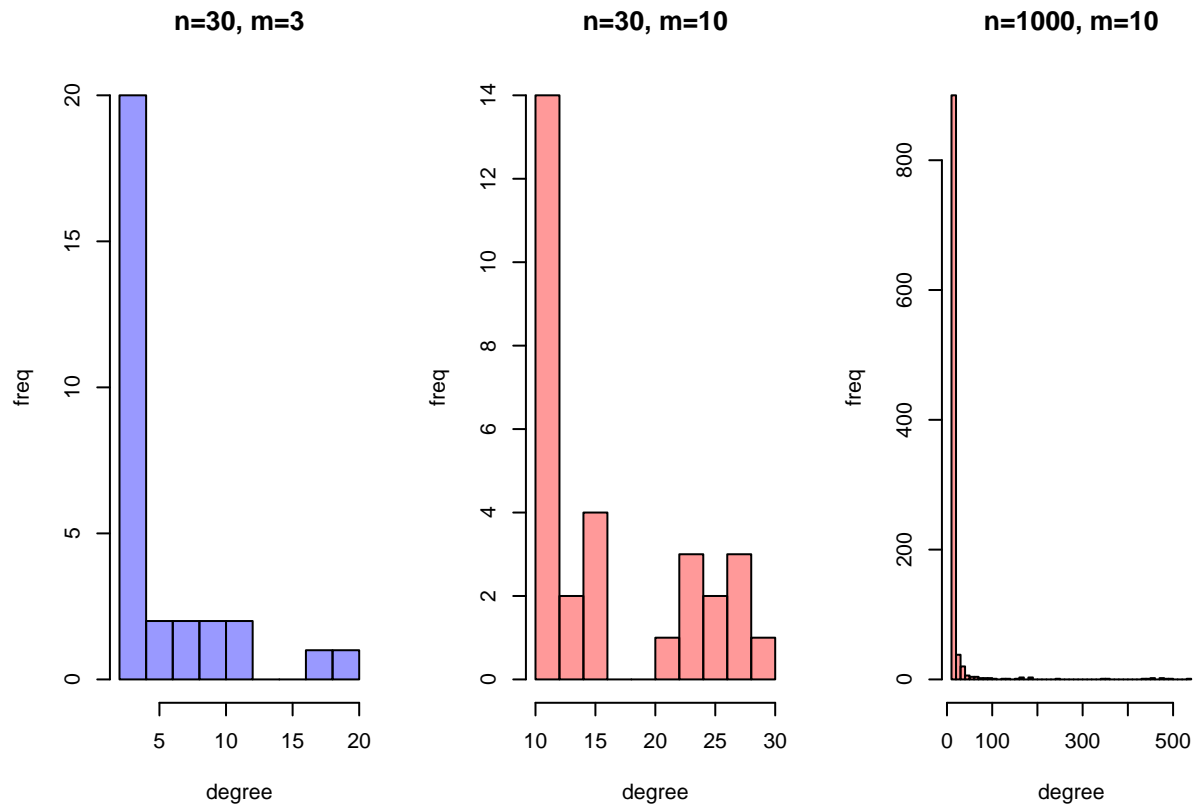
db1 <- degree(b1)
db2 <- degree(b2)
```



```

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] 2.046769
```

Dinamical change

Now, we will look at how clustering coefficient, average path length and node degrees change with 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))}
}

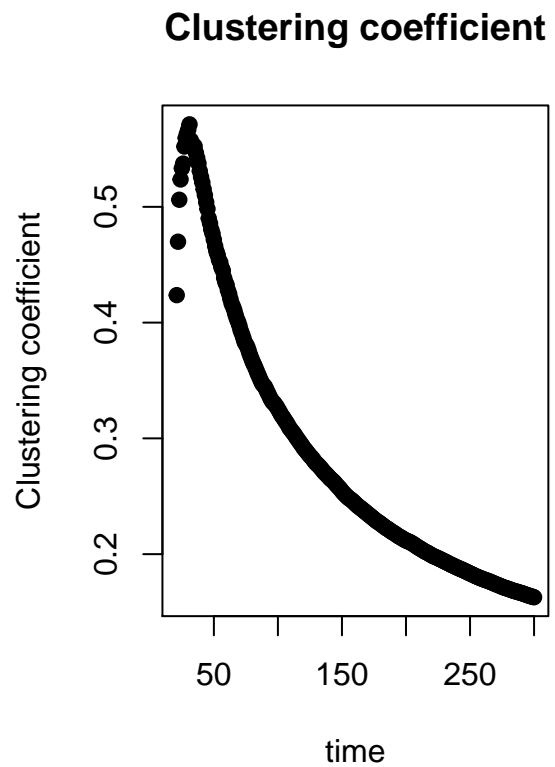
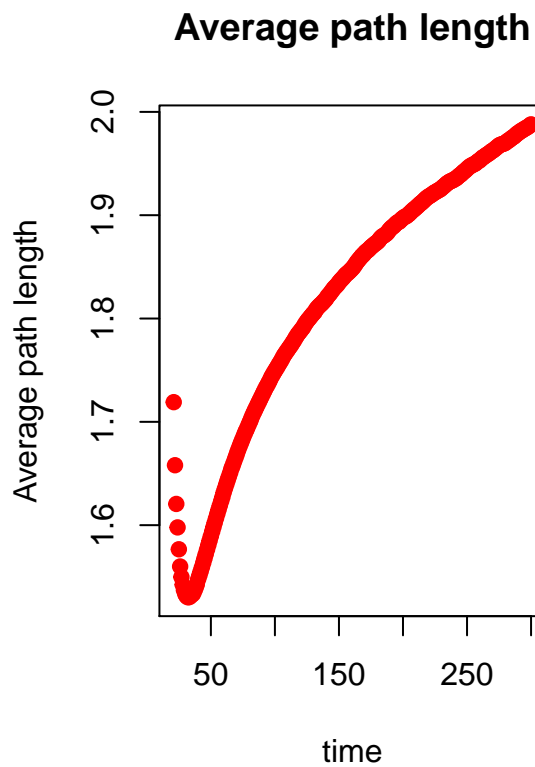
```

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

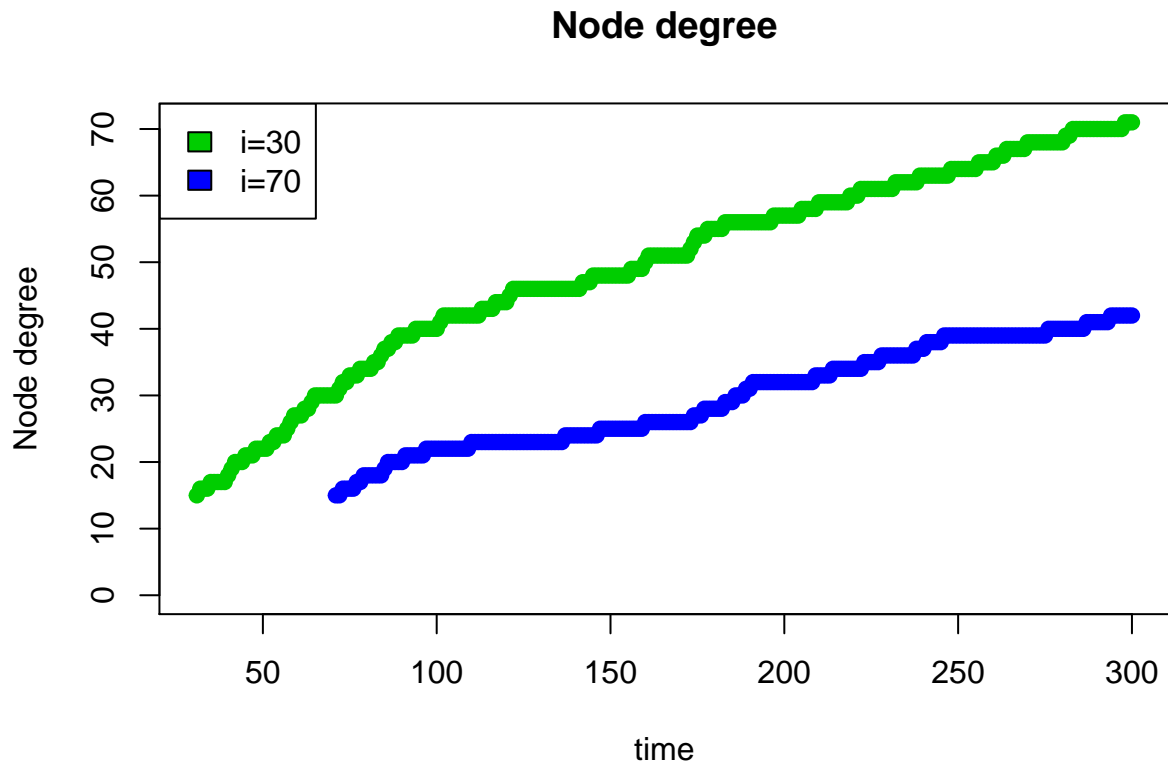
```



```
par(op)
```

And now we may see the node degrees:

```
plot(c(31:300), deg30, pch=19, col=35, ylim=c(0, max(deg30)), xlab="time", ylab="Node degree", main="Node degree")
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 graph
legend("topleft", c("i=30", "i=70"), fill=c(35, 84))
```



Small world model. Watts-Strogats model

We will use a function from igraph package:

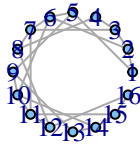
```

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)

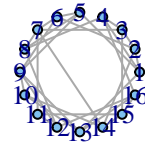
op <- par(mfrow = c(2, 2))
plot(lat, layout=layout.circle(lat), main="Lattice")
plot(ws1, layout=layout.circle(ws1), main="Rewiring probability 0.1")
plot(ws2, layout=layout.circle(ws2), main="Rewiring probability 0.4")
plot(ws3, layout=layout.circle(ws3), main="Rewiring probability 0.9")

```

Lattice



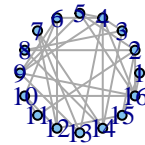
Rewiring probability 0.1



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.